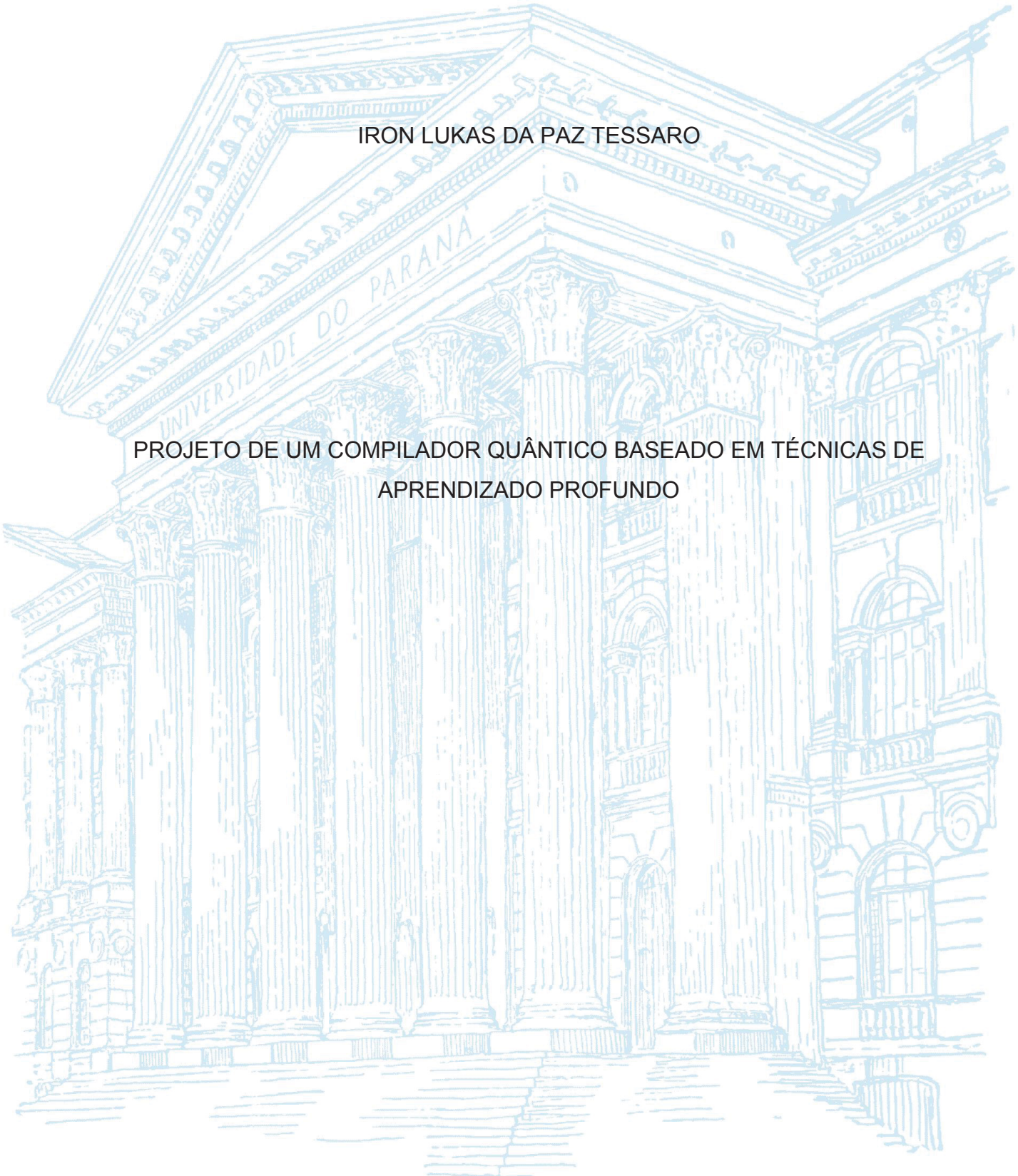
UNIVERSIDADE FEDERAL DO PARANÁ

IRON LUKAS DA PAZ TESSARO



PROJETO DE UM COMPILADOR QUÂNTICO BASEADO EM TÉCNICAS DE APRENDIZADO PROFUNDO

CURITIBA

2019

IRON LUKAS DA PAZ TESSARO

PROJETO DE UM COMPILADOR QUÂNTICO BASEADO EM TÉCNICAS DE APRENDIZADO PROFUNDO

Documento de dissertação de mestrado ao Programa de Pós-Graduação em Engenharia Elétrica – PPGEE da Universidade Federal do Paraná, como requisito à obtenção do grau de Mestre em Engenharia Elétrica, Departamento de Engenharia Elétrica (DELT), da Universidade Federal do Paraná.

Área de concentração: Sistemas Eletrônicos

Orientador: Prof. Leandro dos Santos Coelho

CURITIBA

2019

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

T338p

Tessaro, Iron Lukas
Projeto de um compilador quântico baseado em técnicas de aprendizado profundo [recurso eletrônico] / Iron Lukas Tessaro. – Curitiba, 2019.

Dissertação – Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós – Graduação em Engenharia Elétrica – PPGEE, 2019.

Orientador: Leandro dos Santos Coelho.

1. Computadores quânticos. 2. Computação quântica. 3. Aprendizado do computador. 4. Redes neurais (Computação). I. Universidade Federal do Paraná. II. Coelho, Leandro dos Santos. III. Título.

CDD: 006.3843

Bibliotecária: Vanusa Maciel CRB- 9/1928

TERMO DE APROVAÇÃO

IRON LUKAS DA PAZ TESSARO

PROJETO DE UM COMPILADOR QUÂNTICO BASEADO EM TÉCNICAS DE APRENDIZADO PROFUNDO

Documento de dissertação de mestrado ao Programa de Pós-Graduação em Engenharia Elétrica – PPGEE da Universidade Federal do Paraná, como requisito à obtenção do grau de Mestre em Engenharia Elétrica, Departamento de Engenharia Elétrica (DELT), da Universidade Federal do Paraná.

Orientador:

Prof. Dr. Leandro dos Santos Coelho
Departamento de Engenharia Elétrica, UFPR

Banca examinadora:

Prof. Drª. Viviana Cocco Mariani
Pós-Graduação em Engenharia Mecânica (PPGEM), PUCPR

Prof. Dr. Gideon Villar Leandro
Departamento de Engenharia Elétrica, UFPR

Prof. Dr. Roberto Zanetti Freire
Pós-Graduação em Engenharia de Produção e Sistemas (PPGEPS), PUCPR

Curitiba, 26 de Novembro de 2019.

MINISTÉRIO DA EDUCAÇÃO
SETOR DE TECNOLOGIA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO ENGENHARIA ELÉTRICA - 40001016043P4

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em ENGENHARIA ELÉTRICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **IRON LUKAS DA PAZ TESSARO** intitulada: **Projeto de um compilador quântico baseado em técnicas em técnicas de aprendizado profundo**, sob orientação do Prof. Dr. LEANDRO DOS SANTOS COELHO, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 26 de Novembro de 2019.

LEANDRO DOS SANTOS COELHO
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

VIVIANA COCCO MARIANI
Avaliador Externo (PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ)

GIDEON VILLAR LEANDRO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

ROBERTO ZANETTI FREIRE
Avaliador Externo (PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ)

RESUMO

O computador quântico é um dispositivo que realiza cálculos por meio de propriedades da mecânica quântica, tais como a sobreposição e o entrelaçamento quântico, sendo uma tecnologia promissora na solução de problemas que hoje são inviáveis para os computadores clássicos, tais como a simulação dos estados quânticos de moléculas complexas, a criptografia quântica e a otimização por busca exaustiva de sistemas com centenas de variáveis. Entretanto, o modo peculiar com que a informação é processada e armazenada nos computadores quânticos abre um novo campo de estudos sobre como otimizar a implementação de algoritmos nesses dispositivos para explorar todo o potencial que essa tecnologia pode fornecer. Na concepção de computadores quânticos, assim como nos computadores clássicos, encontra-se a compilação de algoritmos. Por não possuir uma forma determinística de se definir o conjunto de instruções mais eficiente possível que representa um algoritmo em um dado computador quântico, a Compilação quântica é um problema que ainda não possui solução definitiva, sendo um dos focos de pesquisas na área de Computação Quântica. Dado que a compilação quântica pode ser modelada como um problema de busca por um conjunto de portas quânticas que minimize o erro entre o resultado desejado e o resultado da busca desse conjunto, o Aprendizado Profundo foi o grupo de técnicas escolhido para abordar esse problema. Neste contexto, o objetivo dessa dissertação é projetar um compilador quântico utilizando três abordagens de Aprendizado Profundo: rede neural perceptron multicamada profunda, rede neural convolucional e rede “Q” profunda, pelos quais o compilador aprenda as regras de como transformar um algoritmo em instruções de um computador quântico específico. Como resultado deste trabalho, obteve-se um modelo de rede “Q” profunda capaz de compilar algoritmos quânticos que podem ser validados no computador quântico IBM-Q 5 Tenerife por meio de algoritmos de teste, utilizando a métrica de comparação da diferença entre o número de operações quânticas utilizadas pelo compilador quântico desenvolvido e os circuitos ideais que representam esses algoritmos. Demonstra-se, ao final, que técnicas de aprendizado profundo podem ser aplicadas para a solução do problema da compilação quântica, por meio de testes de acurácia.

Palavras-chave: Compilação quântica; Computador quântico; Aprendizado de Máquina; Aprendizado Profundo.

ABSTRACT

The quantum computer is device that performs calculations by using properties of quantum mechanics such as superposition and quantum entanglement and is a promising technology for solving problems that are currently infeasible to be solved by classical computers, such as the simulation of quantum states of complex molecules, quantum cryptography and the brute-force search optimization of system with hundreds of variables. However, the unique way in which information is processed and stored in quantum computers opens up a new field of studies on how to optimize the implementation of algorithm on these devices in order to exploit the full potential that this technology can provide. In the conception of quantum computers, just like in classical computers, lies the compilation of algorithms. Because there is not a deterministic form of defining the most efficient set of instructions that represents an algorithm in a given quantum computer, the Quantum compilation is a problem that does not yet have a definitive solution, being one of the focuses of research in the area of Quantum computing. Since Quantum compilation can be modeled as a problem of searching a set of quantum gates that minimizes the error between a target result and the searching result, Deep learning was the set of techniques chosen to approach this problem. In this context, the goal of this dissertation is to design a quantum compiler by using three deep learning approaches: deep multilayer perceptron neural network, convolutional neural network and deep Q-network, whereby the compiler learns the rules of how to transform an algorithm into instructions for a specific quantum computer. As outcome of this work a trained deep Q-Network model which is able to compile quantum algorithms that can be validated on the IBM-Q 5 Tenerife quantum computer via classic test, by using the metric of comparing the difference between the number of quantum operations used by the developed quantum compiler and the number of quantum operations used in the ideal circuits that represent those algorithms. In the end, it is demonstrated that deep learning techniques can be applied to solve the quantum compilation problem by means of accuracy.

Key-words: Quantum compiling; Quantum computing; Machine learning; Deep learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADAM	–	Estimativa de Momento Adaptativo (do inglês *Adaptive Moment Estimation*)
AM	–	Aprendizado de Máquina (do inglês *Machine Learning*)
ANN	–	Rede Neural Artificial (do inglês *Artificial Neural Network*)
AUTOML	–	Aprendizado de Máquina Automatizado (do inglês *Automated Machine Learning*)
CNN	–	Rede Neural Convolucional (do inglês *Convolutional Neural Network*)
CNOT	–	Porta quântica inversora controlada (do inglês *Controlled Not*)
DQN	–	Rede “Q” Profunda (do inglês *Deep Q-Network*)
DQL	–	Aprendizado “Q” Profundo (do inglês *Deep Q-Learning*)
DRL	–	Aprendizado por Reforço Profundo (do inglês *Deep Reinforcement Learning*)
IBM	–	Corporação de Máquinas de Negócio Internacional (do inglês *International Business Machines Corporation*)
QAQC	–	Compilação Quântica Quanticamente Auxiliada (do inglês *Quantum-Assisted Quantum Compilation*)
MLP	–	Rede neural Perceptron multicamadas (do inglês *Multi-layer Percerptron*)
MAE	–	Erro Médio Absoluto (do inglês *Mean Absolute Error*)
MSE	–	Erro Médio Quadrático (do inglês *Mean Squared Error*)
RELU	–	Unidade Linear Retificada (do inglês *Rectified Linear Unit*)
RF	–	Radiofrequência (do inglês *Radiofrequency*)
SGD	–	Descida de Encosta Estocástica (do inglês *Stochastic Gradient Descent*)

LISTA DE SÍMBOLOS

$\|0\rangle$	–	Vetor de estado quântico para o q-bit 0
$\|1\rangle$	–	Vetor de estado quântico para o q-bit 1
$\|\Psi\rangle$	–	Vetor de estado quântico de um q-bit genérico
$\|\varphi\rangle$	–	Vetor de estado quântico de um q-bit genérico
j	–	Unidade imaginária
m	–	Número de q-bits de um q-bit genérico
n	–	Número de q-bits de um q-bit genérico
α	–	Coeficiente complexo do vetor de estado quântico do q-bit 0
β	–	Coeficiente complexo do vetor de estado quântico do q-bit 1
X	–	Porta lógica quântica de Pauli X
Y	–	Porta lógica quântica de Pauli Y
Z	–	Porta lógica quântica de Pauli Z
H	–	Porta lógica quântica Hadamard
S	–	Porta lógica quântica de fase $\pi/4$
T	–	Porta lógica quântica de fase $\pi/8$
U_{CNOT}	–	Porta lógica quântica inversora controlada
i_n	–	Índice de entradas de um neurônio artificial
z	–	Somatório de entradas ponderadas e constante de um neurônio artificial
w	–	Peso de uma entrada de um neurônio artificial
x	–	Valor de entrada de um neurônio artificial
b	–	Constante de um neurônio artificial
σ	–	Função de ativação
f	–	Valor de saída de uma rede neural artificial
e	–	Número de Euler
o	–	Saída de um neurônio artificial
C	–	Função custo de erro médio quadrático
N	–	Número de elementos de entrada
η	–	Taxa de aprendizado
L	–	Número de camadas de uma rede neural artificial
n_n	–	Índice do neurônio de uma rede neural artificial
l	–	Índice da camada de uma rede neural artificial

v – Operação de convolução

t – Tempo

y – Função de entrada da operação de convolução

h – Função *kernel* da operação de convolução

τ – Variável auxiliar da operação de convolução

i_I – Índice de linha dos *pixels* de uma imagem

j_I – Índice de coluna dos *pixels* de uma imagem

m_K – Índice de linha de um *kernel*

n_K – Índice de coluna de um *kernel*

S – Saída discreta de uma camada de convolução

I – Entrada discreta de uma camada de convolução

K – *Kernel* discreto de uma camada de convolução

O – Limite assintótico superior

s – Estado de um agente

a – Ação de um agente

π – Política de tomada de ação por um agente

r – Recompensa resultante de uma ação de um agente

R – Recompensa total de um episódio

R_t – Recompensa futura total

γ – Fator de desconto de recompensa

Q – Maior recompensa futura

A – Número de arranjos simples

q – Número de q-bits do circuito quântico

p – Número de q-bits afetados pela porta inversora controlada

ε – Somatório de erros absolutos

μ – Elemento da matriz-objetivo

ρ – Elemento da matriz-resposta

Ω – Métrica de pontuação

ϑ – Função *softmax*

Γ – Equação de entropia cruzada

Y – Valor alvo de saída

SUMÁRIO

1 INTRODUÇÃO

Um computador quântico é um dispositivo que faz uso direto de fenômenos quânticos distintos, tais como superposição e entrelaçamento, para processar dados. Os princípios básicos da computação quântica são que as propriedades quânticas podem ser usadas para representar e estruturar dados, e que os mecanismos quânticos podem ser projetados para realizar operações com esses dados. Enquanto os computadores clássicos armazenam dados na forma de bits, computadores quânticos armazenam dados na forma de q-bits, ou bits quânticos. Um q-bit é um sistema quântico de dois estados, e pode ser representado fisicamente como o spin de um único elétron ou a polarização de um único fóton, por exemplo. Em um sistema clássico, um bit precisa estar em um dos dois estados possíveis, normalmente representados como estados 0 e 1. Entretanto, os fenômenos quânticos permitem que um q-bit esteja em uma superposição dos dois estados, ao mesmo tempo, propriedade fundamental para a Mecânica quântica e para a Computação Quântica (NIELSEN; CHUANG, 2010).

Os computadores quânticos reais podem ser acessados por meio da nuvem, como por exemplo, o IBM Q (IBM, 2019), e pesquisadores os têm usado para aprender, conduzir pesquisas e abordar problemas de simulação e otimização de uma forma alternativa à utilizada em computadores clássicos, já que os estados desses problemas podem ser sobrepostos e processados simultaneamente (PORTUGAL *et al.,* 2004). Os computadores clássicos realizam operações e processam informação por meio do modelo padrão da computação, onde toda a informação é reduzida em bits, que podem assumir os valores 0 ou 1, e todo o processamento de dados pode ser realizado via portas lógicas agindo em um ou mais bits ao mesmo tempo. A qualquer momento durante o processamento, os estados de um computador clássico são determinados pelos estados de todos os seus bits, tal que um computador com n bits pode existir em um dos 2^n estados possíveis (ARAÚJO, 2008). Os computadores quânticos também possuem bits mas, ao invés dos valores 0 ou 1, seus bits quânticos, denominados q-bits, podem representar os estados 0, 1, ou uma combinação linear dos dois, propriedade essa denominada de superposição que, por si só, não apresenta nenhuma vantagem já que essa propriedade pode ser obtida por um computador analógico.

Um computador quântico tem a potencial de utilizar um tipo especial de superposição que permite a representação de um número exponencialmente maior de estados lógicos de uma só vez, possuindo também a capacidade de representar superposições entrelaçadas, que são estados do computador que não correspondem a nenhuma associação digital ou analógica dos q-bits individuais (HAN; KIM, 2002). Essas propriedades únicas dos computadores quânticos fazem com que eles sejam mais rápidos em resolver tarefas do que qualquer computador clássico, sejam essas tarefas determinísticas, probabilísticas ou analógicas, que podem ser apresentadas por meio dos algoritmos de Shor (1997) e de Grover (1996), sendo o primeiro um algoritmo para a fatoração de inteiros em números primos e o segundo um algoritmo de busca que, dada uma função caixa preta que retorna um valor distinto para uma das suas possíveis entradas e um valor comum para todas as outras, encontra a entrada que produz esse valor distinto.

Um dos problemas que os computadores quânticos possuem é a perda de informação devido a ruídos no ambiente, denominada decoerência. Uma das maneiras de contornar esse problema é reduzindo a quantidade de operações quânticas por meio de compiladores quânticos (AMY; MOSCA, 2017). Um compilador quântico é um método que adota como entrada um algoritmo e o transforma em uma sequência de portas quânticas (HARROW, 2001). Desde a primeira proposta de um compilador quântico, várias pesquisas no desenvolvimento de compiladores quânticos otimizados foram realizadas, mas até a presente data não foi possível desenvolver um "compilador ótimo", que gere o menor circuito possível para um determinado algoritmo, já que não existe um método determinístico para tal fim (CAMPBELL; HEYFRON, 2018).

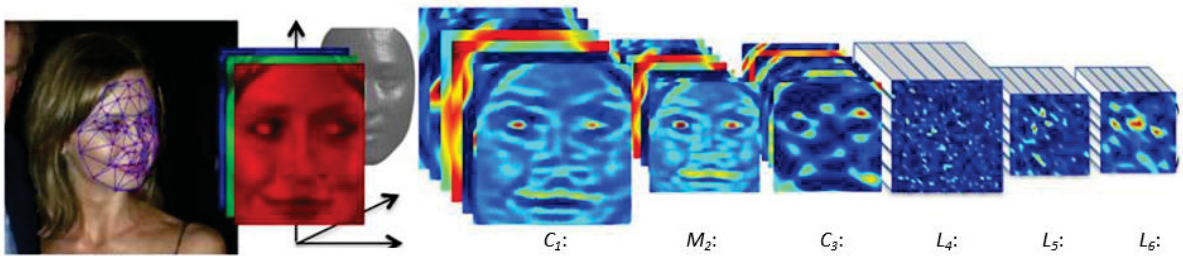
O Aprendizado de Máquina é um campo da Inteligência Artificial que utiliza técnicas estatísticas para permitir que sistemas computadorizados progressivamente melhorem seu desempenho em uma determinada tarefa, por meio de dados, sem serem explicitamente programados para realizar tal tarefa. "Diz-se que um programa aprende com a experiência *E* em relação a alguma classe de tarefas *T* e medida de desempenho *P* se o seu desempenho em tarefas em *T*, medida por *P*, melhora com a experiência *E*." (MITCHEL, 1997).

Um algoritmo que compõe o conjunto de técnicas de Aprendizado de Máquina denominado regressão logística pode, por exemplo, determinar quando uma determinada ação deve ser realizada em um dado ambiente (MOR-YOSEF et al.,

1990). O desempenho de algoritmos de Aprendizado de Máquina depende da representação dos dados que lhe são apresentados. Quando uma abordagem de Aprendizado de Máquina é utilizada para recomendar uma ação, o algoritmo não analisa o conjunto de entradas diretamente. Ao invés disso, o sistema é alimentado por várias partes de informação relevantes desse conjunto de entradas como, por exemplo, o contorno de uma imagem ao invés de cada *pixel* (menor ponto que forma uma imagem digital) dessa imagem. Cada item de informação incluído na representação do conjunto de entradas é conhecido como característica. A abordagem de Aprendizado de Máquina tem mecanismos que podem aprender como cada uma das características do conjunto de entradas se relaciona com a(s) respectiva(s) saída(s), mas não pode influenciar como as características são definidas. Se a entrada do algoritmo for um dado não tratado do ambiente, não é possível fazer predições úteis. Extrair tais características somente alimentando o sistema com o conjunto de entradas não tratadas não é uma tarefa simples, sendo necessário que as técnicas de extração de características do conjunto de entradas sejam definidas a priori e manualmente, podendo limitar a convergência do aprendizado (GOODFELLOW et al., 2016).

O Aprendizado Profundo pode solucionar esse problema introduzindo representações de características que são expressas em termos de outras representações, permitindo que computadores construam conceitos complexos sobre conceitos simples. O Aprendizado Profundo é uma classe de algoritmos de Aprendizado de Máquina que utiliza múltiplas camadas de processamento não linear para a extração de características. No caso da rede neural convolucional, por exemplo, cada camada aprende a transformar seus dados de entrada em uma representação “ligeiramente” mais abstrata e complexa. Em uma aplicação de reconhecimento de imagem, a primeira camada de um modelo de Aprendizado Profundo pode receber uma matriz de *pixels* como entrada e pode aprender a reconhecer as bordas e contrastes na imagem, conforme ilustrado na FIGURA 1.

FIGURA 1 – EXEMPLO DE EXTRAÇÃO DE CARACTERISTICAS - CNN.



FONTE: Adaptado de MARIED (2019).

A segunda camada desse mesmo modelo pode abstrair a identificação de bordas oriundas da primeira camada e aprender a classificar formas geométricas. A terceira camada, por sua vez, pode aprender a classificar objetos a partir das combinações e arranjos dessas formas geométricas, e assim sucessivamente até última camada do modelo, sem que o algoritmo seja programado explicitamente para extrair essas características específicas, o que permite ao modelo extrair características que podem ser incompreensíveis para humanos, mas que resulta em um melhor desempenho do modelo em relação à tarefa em que foi aplicado (GOODFELLOW et al., 2016).

Como exemplo de aplicações de Aprendizado Profundo pode-se citar a RF-Pose de Zhao et al. (2018), uma aplicação de Aprendizado Profundo onde a postura de um humano pode ser estimada através de paredes, utilizando sinais de radiofrequência que as atravessam mas refletem no corpo humano. Por meio de imagens do ambiente e de mapas de calor bidimensionais produzidos pelos sinais de radiofrequência, um modelo formado por redes neurais convolucionais conectadas em paralelo foi treinado, de modo que somente os sinais de radiofrequência sejam necessários para a execução do modelo após o treinamento, sem as imagens de entrada. Assael et al. (2017) apresentaram a LipNet®, um modelo também baseado em redes neurais convolucionais que utiliza imagens sequenciais dos lábios de um locutor como entrada e texto na saída do modelo, aprendendo a fazer leitura labial. Como exemplo de Aprendizado por Reforço Profundo, pode-se citar a OpenAI Five, da empresa OpenAI (2018), que criou um grupo de modelos de redes neurais recorrentes (RNNs), com cada modelo representando um personagem do jogo eletrônico Dota 2, atualmente o esporte eletrônico que possui o torneio mundial com a maior premiação dentre todos os esportes eletrônicos. Esse jogo é disputado por dois times de cinco jogadores cada, onde o objetivo é destruir o núcleo da base

adversária. Utilizando aproximadamente 180 anos em tempo de jogo, com tempo acelerado, os modelos disputam partidas entre si, com cada RNN representando um jogador de um time, e aprendem consigo mesmos como obter uma pontuação melhor, interagindo com as outras RNNs do seu respectivo time.

Considerando o potencial das técnicas de Aprendizado de Máquina, especialmente do subconjunto de Aprendizado Profundo, o escopo deste projeto será desenvolver um compilador quântico baseado nas seguintes técnicas: rede neural perceptron multicamada profunda, rede neural convolucional e rede “Q” profunda, com o objetivo de apresentar uma nova abordagem e um novo método para a solução do problema da compilação de algoritmos quânticos.

1.1 OBJETIVOS

A seguir os objetivos, geral e específicos, desta dissertação são apresentados.

1.1.1 Objetivo geral

O objetivo geral desta dissertação é projetar um compilador quântico por meio de técnicas de Aprendizado Profundo em que, dado uma matriz que represente um determinado algoritmo quântico, o compilador encontre ao menos um circuito quântico que resulte nessa matriz, utilizando somente as portas quânticas disponíveis de um determinado dispositivo quântico.

1.1.2 Objetivos específicos

Os objetivos específicos desta dissertação são os seguintes:

a) Realizar um apanhado da literatura sobre compiladores quânticos e as técnicas utilizadas para isso.
b) Desenvolver um simulador de circuitos quânticos baseado no IBM-Q 5 Tenerife (IBM, 2019).
c) Implementar e treinar uma rede neural artificial perceptron multicamada e uma rede neural convolucional utilizando os dados gerados pelo simulador de circuitos quânticos desenvolvido.

d) Desenvolver e treinar um método de *Q-learning* para provar a viabilidade da técnica para o problema da compilação quântica utilizando um circuito quântico conhecido como teste.
e) Desenvolver uma *Deep Q-Network* utilizando o simulador de circuitos quânticos desenvolvido como ambiente para o processo de decisão de Markov.
f) Comparar os resultados das técnicas utilizando a acurácia dos circuitos encontrados como métrica, onde o resultado dos circuitos compilados é comparado com o resultado dos circuitos de teste.

1.1.3 Contribuições

Como contribuição desta dissertação tem-se a apresentação de um novo método para compilação de circuitos quânticos, por meio de técnicas de Aprendizado de Máquina, especialmente a utilização de Aprendizado por Reforço Profundo, de forma a compilar um algoritmo quântico com a menor quantidade de portas quânticas possíveis. Espera-se que, dado um conjunto de portas quânticas disponíveis em um computador quântico qualquer e um circuito quântico conhecido, o compilador quântico desenvolvido neste trabalho possa encontrar o circuito quântico que gera essa resposta com o mínimo de portas quânticas. Para isso, serão utilizados circuitos conhecidos de teste em sua forma reduzida, onde se sabe a quantidade mínima de portas necessárias.

O resultado desse trabalho pode ser utilizado para a comparação de outras técnicas de compilação quântica e demonstrar que é possível utilizar técnicas de Aprendizado de Máquina para auxiliar otimização de circuitos quânticos. Os simuladores aqui desenvolvidos permitem com que pesquisadores testem e avaliem outras técnicas de Aprendizado de Máquina para essa finalidade, permitindo a evolução contínua dos compiladores quânticos baseados em técnicas de Aprendizado de Máquina, seja diminuindo o custo computacional necessário e/ou otimizando a quantidade de portas quânticas necessária para a compilação.

1.2 METODOLOGIA

O objeto de estudo utilizado foi o dispositivo quântico IBM Q 5 Tenerife, de 5 q-bits (IBM, 2019), pois possui interface online onde é possível realizar simulações de circuitos quânticos e agendar a execução desses circuitos no dispositivo real, permitindo a avaliação dos resultados de forma mais precisa e confiável, já que a simulação não leva em consideração alguns fatores como a decoerência quântica, por exemplo.

Para que fosse possível criar um conjunto de dados com várias amostras de circuitos para o treinamento e testes das técnicas de Aprendizado de Máquina, bem como um ambiente para interação com o agente no Aprendizado por Reforço Profundo, o simulador do dispositivo quântico IBM Q 5 Tenerife foi implementado em ambiente computacional Matlab® R2018a, no sistema operacional Windows® 10. As técnicas de Aprendizado, porém, foram implementadas em linguagem de computação Python 3.5, com o auxílio da plataforma TensorFlow® e a interface de programação de aplicação denominada Keras.

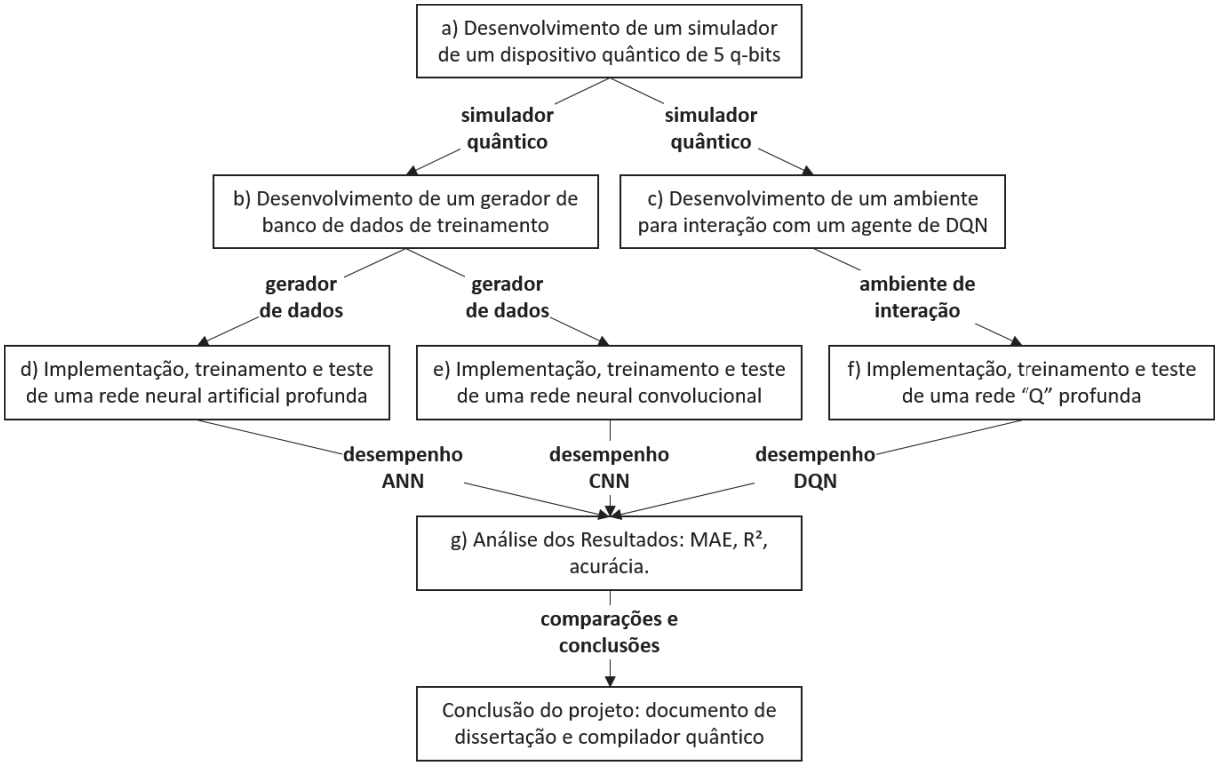
As técnicas de Aprendizado utilizadas foram:

- Redes neurais artificiais (do inglês *Artificial Neural Networks* - ANNs) com perceptron multicamadas, escolhida por ser um método que, na sua configuração mais simples, possui baixo custo computacional para classificação;
- Redes neurais convolucionais (do inglês *Convolutional Neural Networks* - CNNs), escolhida por manter a relação entre os elementos matriciais do conjunto de entradas utilizado, ou seja, leva em consideração a posição dos elementos dos dados de entrada;
- Rede “Q” profunda (do inglês *Deep-Q Network* - DQN), escolhida por não ser limitada à convergência de funções, ou seja, permite mais de uma possibilidade de circuito quântico para o mesmo algoritmo.

1.2.1 Etapas da pesquisa

As etapas de trabalho para o desenvolvimento dessa dissertação são apresentadas na FIGURA 2.

FIGURA 2 – ETAPAS DE DESENVOLVIMENTO DA DISSERTAÇÃO



FONTE: O autor (2019).

a) Desenvolver um simulador de um dispositivo quântico de 5 q-bits – desenvolver um simulador baseado no simulador do dispositivo quântico IBM Q 5 Tenerife, em ambiente computacional Matlab®, contendo as mesmas funcionalidades e o mesmo comportamento, mas com uma interface de entrada diferente, com o objetivo de facilitar a interação das técnicas e das ferramentas utilizadas neste trabalho.
b) Desenvolver um gerador de conjunto de dados de treinamento – com base no simulador de dispositivos quânticos de 5 q-bits, desenvolvido na etapa anterior, desenvolver um gerador de conjunto de dados para treinamento que gera circuitos quânticos aleatórios, testa a validade dos circuitos e salva esses circuitos em um arquivo que pode ser processado diretamente tanto em Matlab® quanto em Python.
c) Desenvolver um ambiente para interação com um agente – utilizando o simulador de dispositivos quânticos, desenvolver um ambiente de tomadas de decisões e recompensas para interação com o agente da técnica DQN.

d) Implementar, treinar e testar uma rede neural artificial profunda – por meio da plataforma Keras e a linguagem de programação Python 3.5, implementar e treinar uma rede neural artificial multicamada utilizando os dados gerados pelo gerador de conjunto de dados desenvolvido.
e) Implementar, treinar e testar uma rede neural convolucional - por meio da plataforma Keras e a linguagem de programação Python, implementar e treinar uma rede neural convolucional utilizando os dados gerados pelo gerador de conjunto de dados desenvolvido.
f) Implementar, treinar e testar uma rede “Q” profunda - por meio da plataforma Keras e a linguagem de programação Python, implementar e treinar uma rede “Q” profunda contendo uma rede neural artificial multicamada utilizando o ambiente desenvolvido.
g) Analisar resultados – comparar os desempenhos das três técnicas de Aprendizado Profundo e avalia-las entre si, utilizando a acurácia dos circuitos encontrados como métrica, onde o resultado dos circuitos compilados é comparado com o resultado dos circuitos de teste.

1.2.2 Organização do documento de dissertação

Este documento de dissertação está dividido em seis capítulos. No Capítulo 1 foi apresentada uma contextualização do problema da compilação quântica e da utilização de técnicas de Aprendizado de Máquina para aplicações semelhantes.

O Capitulo 2 apresenta uma descrição sobre a computação quântica e a matemática necessária para o seu entendimento.

O Capitulo 3 detalha os conceitos sobre técnicas de Aprendizado de Máquina utilizadas para a modelagem de um compilador quântico.

O Capitulo 4 apresenta detalhes das ferramentas desenvolvidas para o auxílio da geração dos conjuntos dados de entrada e saída, o desenvolvimento do simulador do dispositivo quântico e o ambiente de interação para o agente da técnica DQN, bem como os ajustes dos parâmetros realizados nas técnicas de Aprendizado Profundo.

O Capitulo 5 aborda os resultados obtidos para as técnicas ANN, CNN e DQN aplicadas em quatro configurações diferentes de circuitos quânticos.

Por fim, o Capitulo 6 discorre sobre as conclusões obtidas por meio da análise dos resultados obtidos no Capítulo 5.

2 FUNDAMENTOS BÁSICOS E REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados os conceitos sobre computação quântica e em sequência uma coletânea dos trabalhos recentes, relacionados ao tema desta dissertação.

2.1 COMPUTAÇÃO QUÂNTICA

Em analogia aos conceitos de computação clássica, na computação quântica existe uma quantidade mínima de informação que pode ser representada, sendo denominados de q-bit ou bit quântico. A principal diferença entre os bits clássicos e os q-bits é que ao invés do segundo ser representado por apenas dois valores, 1 ou 0, sua definição se dá por dois vetores de estado, $|0\rangle$ e $|1\rangle$, que contêm as seguintes informações:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{e} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.1}$$

Essa notação, conhecida como *Bracket*, é utilizada em mecânica quântica e foi descrita inicialmente pelo físico britânico Paul Dirac (ARAÚJO, 2008). Um q-bit genérico $|\psi\rangle$, diferentemente de um bit genérico, é representado por uma combinação linear dos dois vetores de estados descritos em (2.1), tal que:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.2}$$

onde $|0\rangle$ e $|1\rangle$ formam uma base canônica, $|\psi\rangle$ é uma superposição dos vetores e α e β são números complexos tais quais:

$$\alpha = a + jb \quad \text{e} \quad \beta = c + jd, \tag{2.3}$$

onde a e c são as partes reais de α e β, e b e d são as suas partes imaginárias, respectivamente. Pode-se verificar que um q-bit pertence ao domínio $\Re^4$, portanto sua regra de normalização segue a regra dos elementos contidos no mesmo, ou seja:

$$a^2 + b^2 + c^2 + d^2 = 1. \tag{2.4}$$

O que significa que:

$$|\alpha|^2 + |\beta|^2 = 1, \tag{2.5}$$

onde $|\alpha|^2$ é a probabilidade do q-bit observado estar no estado $|0\rangle$ e $|\beta|^2$ é a probabilidade do mesmo estar no estado $|1\rangle$.

A interpretação física do q-bit é que ele se encontra ao mesmo tempo tanto no estado $|1\rangle$ quanto no $|0\rangle$, podendo armazenar uma quantidade infinita de informações no seu vetor de superposições, ou seja, $|\psi\rangle$. Porém, devido a essa informação estar em um nível quântico, quando há a necessidade de observar o estado atual do q-bit, o mesmo entra em colapso e assume um estado único, determinado pelas probabilidades $|\alpha|^2$ e $|\beta|^2$ correspondentes (HAN; KIM, 2002).

Para que possa ser definida a utilização de diversos q-bits em conjunto para a representação de uma quantidade maior de estados, Portugal *et al.* (2004) propuseram a utilização de produtos tensoriais entre os q-bits desejados. Um produto tensorial entre dois estados diferentes:

$$|\psi\rangle = \begin{bmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_m \end{bmatrix} \quad \text{e} \quad |\varphi\rangle = \begin{bmatrix} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_n \end{bmatrix},$$

é denotado por $|\psi\rangle \otimes |\varphi\rangle$ e gera um estado $|X\rangle$ como resultado, que é demonstrado na sequência:

$$|X\rangle = \begin{bmatrix} \psi_1\varphi_1 \\ \psi_1\varphi_2 \\ \vdots \\ \psi_1\varphi_n \\ \psi_2\varphi_1 \\ \psi_2\varphi_2 \\ \vdots \\ \psi_2\varphi_n \\ \vdots \\ \psi_m\varphi_1 \\ \psi_m\varphi_1 \\ \vdots \\ \psi_m\varphi_n \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \alpha_{2^m-1} \\ \alpha_{2^m} \end{bmatrix}, \tag{2.6}$$

onde $\psi_i\varphi_{j1}$ é o produto usual entre os números complexos.

O produto tensorial entre dois q-bits resulta em um estado genérico $|\psi\rangle$, quando da superposição de quatro estados, $|00\rangle$, $|01\rangle$, $|10\rangle$ e $|11\rangle$, ou seja:

$$|\psi\rangle = \alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle, \tag{2.7}$$

onde a amplitude dos coeficientes complexos é dada por:

$$\sum_{i=1}^{4} |\alpha_i|^2 = 1. \tag{2.8}$$

Observa-se que, para um produto tensorial de m q-bits, existirão 2^m estados disponíveis. Generalizando então, para casos com m q-bits, assumindo x_i como a representação de um estado, pode-se afirmar que:

$$|X\rangle = \sum_{i=1}^{2^m} \alpha_i x_i, \tag{2.9}$$

onde as amplitudes dos estados devem satisfazer a seguinte equação:

$$\sum_{i=1}^{2^m} |\alpha_i|^2 = 1 \quad . \tag{2.10}$$

Outro conceito importante na computação quântica é o conceito de emaranhamento, o qual define que um estado de 2 q-bits nem sempre é o resultado do produto tensorial entre dois q-bits. Pode-se tomar como exemplo um estado $|01\rangle$, que pode ser escrito como um produto tensorial dos estados $|0\rangle$ e $|1\rangle$, conforme segue:

$$|01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} .$$

No entanto, o estado:

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

é dito estar em um estado de entrelaçamento quântico, pois não existe produto tensorial entre estados de um q-bit que resulte nele.

Para que seja possível realizar operações e evoluções nos q-bits sem perder o foco probabilístico representado pela equação (2.5), existem as portas quânticas. Elas podem ser representadas tanto matematicamente (sendo considerada uma operação unitária U) quanto logicamente (utilizando circuitos quânticos).

Existem vários tipos de portas quânticas utilizadas para transformar um q-bit genérico $|\psi\rangle$ em um outro estado $U|\psi\rangle$, tais quais a porta NOT, a *Controlled* NOT (CNOT), a porta T, a porta S, a porta H e a porta de rotação de Han e Kim (2002).

A porta NOT quântica é representada por um operador unitário X que deve, analogamente ao caso clássico, transformar estado 0 em 1 e vice-versa, satisfazendo as seguintes condições:

$$X|0\rangle = |1\rangle \quad \text{e} \quad X|1\rangle = |0\rangle . \tag{2.11}$$

Para satisfazer a equação (2.11), a matriz unitária X deve ser composta da seguinte maneira:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{2.12}$$

Quando essa porta é aplicada aos estados $|0\rangle$ e $|1\rangle$, o resultado é o seguinte:

$$\begin{aligned} X|0\rangle &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle \\ X|1\rangle &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle \end{aligned} . \tag{2.13}$$

Quando a porta NOT é aplicada a um q-bit genérico $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, assim como a porta NOT clássica, gera a seguinte resposta:

$$X|\psi\rangle = \beta|0\rangle + \alpha|1\rangle . \tag{2.14}$$

A porta CNOT quântica é uma operação definida sobre pelo menos dois q-bits, um de controle e um alvo. Uma porta controlada deve ser ativada apenas se o q-bit de controle estiver no estado $|1\rangle$, modificando o valor do segundo q-bit. Numa porta CNOT quântica, os q-bits podem estar superpostos e também emaranhados. A representação da porta CNOT quântica em formato matricial, para 2 q-bits, é a seguinte:

$$U_{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.15}$$

Sua aplicação aos quatro estados do produto tensorial entre dois q-bits, considerando que o primeiro é o de controle, é:

$$U_{CNOT}\left|00\right\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$U_{CNOT}\left|01\right\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$U_{CNOT}\left|10\right\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

$$U_{CNOT}\left|11\right\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}. \qquad (2.16)$$

A porta Hadamard, ou porta H, é importante para a aplicação do algoritmo de Grover (PORTUGAL *et al.*, 2004). Tal porta é definida pelo operador:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \qquad (2.17)$$

Aplicando a porta H aos estados $\left|0\right\rangle$ e $\left|1\right\rangle$, é obtido:

$$H\left|0\right\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} = \frac{1}{\sqrt{2}} (\left|0\right\rangle + \left|1\right\rangle)$$

$$H\left|1\right\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix} = \frac{1}{\sqrt{2}} (\left|0\right\rangle - \left|1\right\rangle). \qquad (2.18)$$

O resultado encontrado demonstra que a porta H transforma um estado de um q-bit em uma superposição dos dois estados possíveis, com uma probabilidade de 50% de se obter qualquer um quando uma medição for realizada.

Para o caso genérico, com m q-bits , pode-se afirmar que:

$$H^{\otimes m}|0\ldots0\rangle = (H|0\rangle)^{\otimes m}$$

$$H^{\otimes m}|0\ldots0\rangle = (\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))^{\otimes m}$$

$$H^{\otimes m}|0\ldots0\rangle = \frac{1}{\sqrt{2^m}}\sum_{i=1}^{2^m}|i\rangle \quad . \tag{2.19}$$

A porta de rotação foi apresentada por Han e Kim (2002), para atualizar a probabilidade dos indivíduos (q-bits) serem observados em um estado $|0\rangle$ ou $|1\rangle$ em um algoritmo genético, sendo definida por:

$$U(\Delta\theta_i) = \begin{bmatrix} \cos(\Delta\theta_i) & -sen(\Delta\theta_i) \\ sen(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix}, \tag{2.20}$$

onde $\Delta\theta_i, i = 1,2,3,\ldots,m$ é o ângulo da rotação de cada um dos m q-bits em direção ao estado $|0\rangle$ ou $|1\rangle$, dependendo de seu sinal. Segue na sequência uma aplicação dessa porta a um q-bit genérico $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, com $\Delta\theta_i = \pi/2$:

$$U(\pi/2)|\psi\rangle = \begin{bmatrix} \cos(\pi/2) & -sen(\pi/2) \\ sen(\pi/2) & \cos(\pi/2) \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} -\beta \\ \alpha \end{bmatrix}$$

$$|\psi\rangle = -\beta|0\rangle + \alpha|1\rangle \quad . \tag{2.21}$$

2.2 TRABALHOS RELACIONADOS

Neste capítulo são apresentados alguns trabalhos realizados ao longo dos anos com o objetivo de determinar algoritmos para a compilação de circuitos quânticos a partir de um conjunto finito de operadores (portas) quânticos.

A técnica para elaboração de circuitos quânticos, apresentada por Barenco et al. (1995), define propriedades genéricas para sistemas formados por qualquer quantidade de q-bits. Nesse trabalho, é comentado que existe um conjunto finito de operadores lineares que são universais, chamado de conjunto universal de portas quânticas. Mesmo com o método apresentado, o fato de haver mais de uma operação linear que leva ao mesmo resultado faz com que haja variáveis livres nesse método que precisam ser escolhidas arbitrariamente e, mesmo com restrições do método limitando as escolhas, não permite um método direto e determinístico que gere o melhor resultado.

Harrow (2001) ressaltou a importância da otimização de compiladores quânticos dentro de todo o campo da computação quântica e apresentou uma maneira de compilar algoritmos quânticos utilizando o teorema de Solovay-Kitaev, teorema esse que demonstra a possibilidade de obter uma aproximação de qualquer operação quântica desejada usando sequências relativamente curtas de portas quânticas do conjunto universal (KITAEV, 1997). Dawson e Nielsen (2006) apresentaram um algoritmo, chamado de Solovay-Kitaev, baseado na tese de Harrow (2001), para a decomposição de uma porta quântica de um único q-bit em uma sequência de portas universais de modo eficiente. Também é apresentado uma generalização do método para múltiplos q-bits.

Segundo Markov e Saeedi (2012), a redução de tamanho de circuitos quânticos contribui para a redução de recursos necessários para computadores quânticos de maneira não linear, pois circuitos maiores implicam em uma sobrecarga para a correção de erro quântico. Em comparação com a fatoração clássica de números naturais em números primos, circuitos menores podem tornar os computadores quânticos mais competitivos, além de permitir uma execução mais rápida de algoritmos quânticos simulados em computadores clássicos.

Duclos-Cianci e Poulin (2015) apresentaram um método que une dois estágios para se obter um método de tolerância à falha na computação quântica. O primeiro dos estágios é a geração de um conjunto universal discreto de operações tolerantes à falha por meio de transformações de correção de erro e o refinamento de estados. O segundo estágio é a síntese de transformações arbitrárias em um circuito quântico, por meio da combinação dos elementos obtidos no primeiro estágio. Essas combinações no segundo estágio substituem a compilação quântica, mas dependem

do conjunto de operações do primeiro estágio, que podem não fazer parte do conjunto de portas quânticas disponíveis em um computador quântico real.

Amy e Mosca (2017) também ressaltaram a importância de compiladores quânticos mais eficientes devido ao efeito que o ruído em computações quânticas tem. Segundo eles, muitos pesquisadores da área de computação quântica, que trabalham com métodos de tolerância à falha, passaram a focar na otimização de compiladores ao invés da otimização de circuitos físicos para a mitigação do efeito de ruído, por questões de custo.

Shahri (2017) apresentou um novo método para implementar uma operação da porta CNOT quântica para um sistema com dois ou mais q-bits, por meio de uma rede neural artificial. Inicialmente, essa rede neural artificial foi treinada utilizando a retropropagação do erro como método de atualização dos ganhos e a descida de encosta como otimizador, desconsiderando o acoplamento dos q-bits. Após isso, um método de força bruta foi adicionado e que auxiliou o algoritmo de otimização a atingir um erro médio quadrático menor.

Campbell e Heyfron (2018) abordaram o problema de múltiplas decomposições de algoritmos em circuitos quânticos para um mesmo algoritmo, enfatizando a redução da contagem de uma porta quântica universal chamada de porta T. Segundo eles, uma métrica importante para definir se um circuito é mais eficiente que o outro é por meio da contagem de portas T. Para um sistema simples de um único q-bit, a decomposição do circuito é um problema essencialmente solucionado, mas para sistemas de múltiplos q-bits, a decomposição de algoritmos em circuitos quânticos torna-se uma tarefa mais difícil, e requerem um tempo de execução clássico exponencial. Neste trabalho eles apresentam um compilador que foi capaz de reduzir a contagem de portas T em 97%, com uma média de economia de delas de 20%, quando comparados com os melhores compiladores quânticos até então.

Khatri et al. (2019) apresentaram uma metodologia para compilação quântica chamada de compilação quântica quanticamente auxiliada (QAQC), onde um computador quântico fornece uma aceleração computacional exponencial na avaliação do custo de um circuito quântico, ou seja, na avaliação de quão bem um circuito gerado se aproxima de um circuito desejado. A princípio, o QAQC permite a compilação de algoritmos maiores que os métodos clássicos utilizados para compilação quântica devido ao aumento da velocidade de processamento

exponencial fornecido pelo uso de um computador quântico. Como teste dos princípios do QAQC, Khatri et al. (2019) implementaram os conceitos desenvolvidos aos computadores quânticos da IBM (2019) e da Rigetti (2019) para compilar diversos circuitos de 1 q-bit em suas portas quânticas nativas, sendo a primeira vez em que um computador quântico foi utilizado para compilar um circuito de 1 q-bit. O QAQC também foi aplicado em um simulador com e sem ruído, para circuitos unitários de 9 q-bits.

Os principais resultados obtidos no trabalho desenvolvido por Khatri et al. (2019) foram os seguintes. Primeiro, eles escolheram uma função custo que envolvia as diferenças entre os resultados entre um circuito quântico gerado e um conhecido, e foi provado que isso satisfazia quatro critérios: que era confiável, que era eficiente para ser calculado por um computador quântico, que possuía um significado operacional e que escalava de acordo com o tamanho do problema. Segundo, eles apresentaram circuitos com poucas portas quânticas que eram capazes de calcular a função custo em um computador quântico. Terceiro, eles provaram que o cálculo da função custo apresentada possui uma complexidade quântica que não pode ser eficientemente calculada por um computador clássico, sob premissas de complexidade viáveis. Isso estabeleceu uma prova da dificuldade da implementação do método QAQC, para fins de simulação, em computadores clássicos, sendo os testes de parâmetros reais e ruidosos realizados no computador quântico IBM-Q 5 Tenerife (IBM, 2019).

Como contribuição geral, o QAQC é apresentado como um algoritmo que faz outros algoritmos sejam mais eficientes quando implementados, já que a quantidade de operações quânticas pode ser reduzida, levando à diminuição de problemas relacionados com a decoerência quântica gerada em circuitos muito grandes, permitindo o aumento de aplicações possíveis para computadores quânticos.

3 FUNDAMENTOS DE APRENDIZADO DE MÁQUINA

Neste capítulo são abordados os fundamentos das técnicas de regressão da área de Aprendizado de Máquina denominadas ANN, CNN e DQN.

3.1 APRENDIZADO DE MÁQUINA

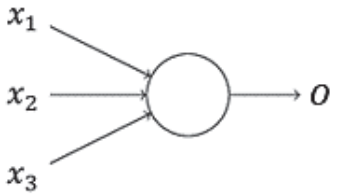
A área de Aprendizado de Máquina estuda métodos de como construir programas de computador que melhorem seus desempenhos em uma determinada tarefa com a experiência. Aplicações diversas têm sido desenvolvidas utilizando técnicas de Aprendizado de Máquina, desde programas de mineração de dados que aprendem a detectar transações de cartão de crédito fraudulentas (MITCHEL, 1997) até a detecção da postura de um humano através de paredes, utilizando sinais de radiofrequência que as atravessam mas refletem no corpo humano (ZHAO et al., 2018).

As seções a seguir apresentam as técnicas de Aprendizado de Máquina que serão utilizadas neste trabalho, começando pelas redes neurais artificiais e como a base de conhecimento deste algoritmo é formada, seguida das redes neurais convolucionais, algoritmo que faz parte do subconjunto Aprendizado Profundo, e da técnica de Aprendizado por Reforço denominada *Q-learning*. Por fim, é apresentada a técnica que utiliza as redes neurais juntamente com o *Q-learning* denominada *Deep Q-Learning* (DQN).

3.1.1 Redes neurais artificiais

Segundo Nielsen (2015), uma rede neural artificial é formada pela conexão de dois ou mais elementos matemáticos chamados neurônios artificiais. Esses neurônios funcionam como funções que recebem diversas entradas e produzem uma única saída, como apresentado na FIGURA 3.

FIGURA 3 – MODELO GERAL DE UM NEURÔNIO ARTIFICIAL



FONTE: Adaptado de NIELSEN (2015).

Para cada entrada x_i do neurônio artificial, um peso w_i é atribuído, assim como uma constante denominada polarização (*bias*), b, para o neurônio. As entradas ponderadas e a constante são somadas, resultando em z, sendo definida por:

$$z = \sum_{i_n} w_{i_n} x_{i_n} + b. \tag{3.1}$$

Esse somatório ponderado de entradas com a constante é válido para todos os modelos de neurônios artificiais. O que difere um tipo de neurônio de outro é a chamada função de ativação. A função de ativação é responsável por transformar a equação linear (3.1) em uma equação não-linear, fazendo com que uma rede neural possa representar qualquer tipo de função (NIELSEN, 2015). Existem diversos tipos de funções de ativação, sendo a função sigmoide a mais comum entre elas:

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{3.2}$$

Inserindo a equação (3.1) na equação (3.2), obtém-se a saída do neurônio artificial, com função de ativação sigmoide, por meio da equação (3.3):

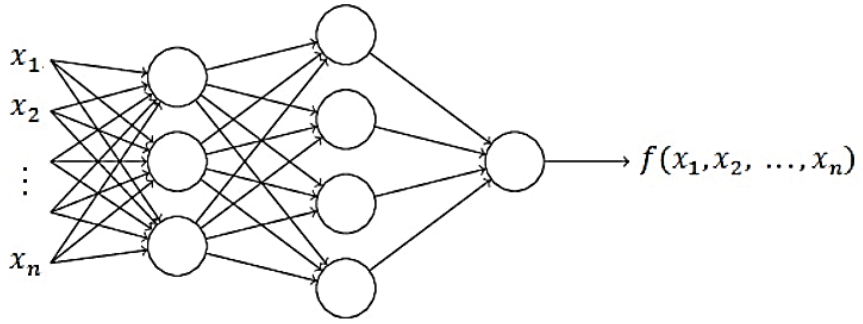
$$o = \frac{1}{1 + e^{-(\sum_{i_n} w_{i_n} x_{i_n} + b)}}. \tag{3.3}$$

A representação de uma função $f(x_1, x_2, \ldots, x_n)$ por um neurônio artificial é dada ajustando-se os pesos de forma que, para um dado conjunto de entradas $\{x_1, x_2, \ldots, x_n\}$, o seja aproximadamente igual a $f(x_1, x_2, \ldots, x_n)$ (NIELSEN, 2015).

Um neurônio somente não é suficiente para representar qualquer tipo de função. É necessário agrupar vários deles em forma de camadas, e conectar camadas

entre si, constituindo uma rede neural multicamada, o que aumenta o número de variáveis (pesos) a serem ajustados e a complexidade das funções intermediárias dentro da própria rede (NIELSEN, 2015). Considerando o modelo do neurônio artificial da FIGURA 3, apresenta-se uma rede neural multicamada na FIGURA 4.

FIGURA 4 – MODELO GERAL DE UMA REDE NEURAL MULTICAMADA



FONTE: Adaptado de NIELSEN (2015).

Não existe, até o momento, uma regra geral de quantas camadas e quantos neurônios em cada camada são necessários para obter um melhor resultado, já que isso depende dos dados disponíveis (NIELSEN, 2015), apesar de ser possível a utilização de algoritmos de otimização para encontrar a melhor combinação desses parâmetros, como fazem os algoritmos de AutoML, onde os hiperparâmetros são escolhidos automaticamente (GIJSBERS et al., 2019). Um método normalmente utilizado é o de construir a primeira camada com o número de neurônios igual ao número de variáveis de entrada do modelo e a saída, obrigatoriamente, com o número de neurônios igual ao número de variáveis de saída. A partir desse ponto, trabalha-se empiricamente com as chamadas camadas ocultas (isto é, camadas entre as camadas de entrada e saída), sempre procurando o menor número possível de elementos (GOODFELLOW et al., 2016).

O termo “aprender” para uma rede neural significa ajustar os pesos e constantes de todos os neurônios para que ela consiga representar uma função que relaciona dados de entrada com dados de saída. A técnica mais utilizada para realizar o ajuste dos pesos e constantes é o algoritmo de retropropagação do erro juntamente com algum método de otimização (NIELSEN, 2015).

3.1.1.1 Descida de encosta

Segundo Ruder (2017), a descida de encosta é um dos mais populares algoritmos de otimização e de longe a forma mais comum de otimizar uma rede neural artificial. Esse algoritmo se baseia no gradiente do espaço de busca para encontrar um mínimo local, incrementando passos proporcionalmente ao gradiente. Esses passos são aplicados a uma determinada função custo que será minimizada. No caso de redes neurais artificiais, o mais comum é utilizar o erro médio quadrático (do inglês *mean squared error*, MSE) como função custo, definida pela equação (3.4). Nessa equação, $\boldsymbol{w}$ é o conjunto de todos os pesos da rede, $\boldsymbol{b}$ é o conjunto de todas as constantes dos neurônios, N é o número de amostras de entrada, $\boldsymbol{o}$ é o vetor de saídas da rede neural quando $\boldsymbol{x}$ é a entrada e $f(\boldsymbol{x})$ é o vetor de saídas esperadas (NIELSEN, 2015), tal que

$$C(\boldsymbol{w},\boldsymbol{b}) = \frac{1}{N}\sum_{\boldsymbol{x}}[f(\boldsymbol{x}) - \boldsymbol{o}(\boldsymbol{w},\boldsymbol{b},\boldsymbol{x})]^2. \tag{3.4}$$

O valor ideal do erro médio quadrático, ao final do treinamento da rede neural artificial, é nulo, pois isso significa que $f(\boldsymbol{x})$ e $\boldsymbol{o}(\boldsymbol{w},\boldsymbol{b},\boldsymbol{x})$ são iguais (isto é, a saída da rede é igual à saída esperada). Supondo $C(\boldsymbol{w},\boldsymbol{b})$ como a função a ser otimizada pelo algoritmo descida de encosta, define-se a variação da função $C(\boldsymbol{w},\boldsymbol{b})$ em relação a uma variação em $\boldsymbol{w}$ e $\boldsymbol{b}$ de acordo com a equação (3.5) (NIELSEN, 2015), tal que

$$\Delta C(\boldsymbol{w},\boldsymbol{b}) \approx \frac{\partial C}{\partial \boldsymbol{w}}\Delta\boldsymbol{w} + \frac{\partial C}{\partial \boldsymbol{b}}\Delta\boldsymbol{b}. \tag{3.5}$$

Para minimizar $C(\boldsymbol{w},\boldsymbol{b})$, sua variação deve ser negativa. Define-se então o gradiente de $C(\boldsymbol{w},\boldsymbol{b})$ na equação (3.6) onde

$$\nabla C(\boldsymbol{w},\boldsymbol{b}) = \left\langle \frac{\partial C}{\partial \boldsymbol{w}}, \frac{\partial C}{\partial \boldsymbol{b}} \right\rangle. \tag{3.6}$$

Reescrevendo a equação (3.5) utilizando a equação (3.6), e reescrevendo $\boldsymbol{w}$ e $\boldsymbol{b}$ como coordenadas de um vetor, obtém-se a equação (3.7) tal que

$$\Delta C(\boldsymbol{w}, \boldsymbol{b}) \approx \nabla C(\boldsymbol{w}, \boldsymbol{b}) \cdot \Delta\langle \boldsymbol{w}, \boldsymbol{b}\rangle^T. \quad (3.7)$$

Para fazer com que $\Delta C(\boldsymbol{w}, \boldsymbol{b})$ seja sempre negativo, o vetor formado por $\boldsymbol{w}$ e $\boldsymbol{b}$ na equação (3.7) deve ser igual à

$$\Delta\langle \boldsymbol{w}, \boldsymbol{b}\rangle^T = -\eta \nabla C(\boldsymbol{w}, \boldsymbol{b})^T. \quad (3.8)$$

O termo η é denominado taxa de aprendizagem quando se trata de redes neurais artificiais. Substituindo a equação (3.8) na equação (3.7), e restringindo a taxa de aprendizagem como sendo sempre maior que zero, é garantida a convergência do algoritmo, já que a variação da função $C(\boldsymbol{w}, \boldsymbol{b})$ é sempre menor ou igual a zero, tal que

$$\Delta C(\boldsymbol{w}, \boldsymbol{b}) \approx \nabla C(\boldsymbol{w}, \boldsymbol{b}) \cdot [-\eta \nabla C(\boldsymbol{w}, \boldsymbol{b})^T] = -\eta \|\nabla C(\boldsymbol{w}, \boldsymbol{b})\|^2. \quad (3.9)$$

Sendo $\boldsymbol{w}'$e $\boldsymbol{b}'$ como os valores atualizados das variáveis $\boldsymbol{w}$ e $\boldsymbol{b}$, define-se:

$$\langle \boldsymbol{w}', \boldsymbol{b}'\rangle^T - \langle \boldsymbol{w}, \boldsymbol{b}\rangle^T = \Delta\langle \boldsymbol{w}, \boldsymbol{b}\rangle^T = -\eta \nabla C(\boldsymbol{w}, \boldsymbol{b})^T. \quad (3.10)$$

Por fim, rearranjando a equação (3.10), encontra-se o método de atualização das variáveis independentes da função $C(\boldsymbol{w}, \boldsymbol{b})$, representando pela equação (3.11), de modo a garantir que seu valor nunca aumente caso a equação (3.8) seja respeitada (isto é, a variação dos pesos e constantes tenda à zero) (RUDER, 2017).

$$\langle \boldsymbol{w}', \boldsymbol{b}'\rangle = \langle \boldsymbol{w}, \boldsymbol{b}\rangle - \eta \nabla C(\boldsymbol{w}, \boldsymbol{b}). \quad (3.11)$$

3.1.1.2 Retropropagação do erro

Segundo Nielsen (2015), para determinar o gradiente da equação (3.4) é necessário calcular sua derivada parcial em relação a todos os pesos e constantes existentes na rede, para depois atualizá-los com a equação (3.11). O procedimento para obter o gradiente consiste em propagar diretamente as entradas do conjunto de treinamento, calcular o erro entre a saída da rede e do conjunto de treinamento, utilizando a equação (3.4), e calcular as derivadas parciais do erro em relação a cada

peso e constante da rede. Como as variações dos erros das camadas posteriores influenciam as variações das camadas anteriores, na forma de uma cadeia de derivadas (regra da cadeia), o cálculo é realizado da saída para a entrada, pois as derivadas parciais das camadas posteriores não precisam ser recalculadas. Sendo L o número de camadas da rede neural, $\boldsymbol{o}^l$ o vetor de saída da camada l e $\boldsymbol{z}^l$ o vetor das somas ponderadas de cada camada, define-se a derivada do erro em relação à soma ponderada da camada l, de forma que

$$\frac{\partial C}{\partial \boldsymbol{z}^l} = \frac{\partial C}{\partial \boldsymbol{o}^L} \cdot \frac{\partial \boldsymbol{o}^L}{\partial \boldsymbol{z}^L} \cdot \left(\prod_{n_n=1}^{L-l} \frac{\partial \boldsymbol{z}^{L-n_n+1}}{\partial \boldsymbol{o}^{L-n_n}} \cdot \frac{\partial \boldsymbol{o}^{L-n_n}}{\partial \boldsymbol{z}^{L-n_n}} \right). \tag{3.12}$$

As derivadas parciais do erro com relação aos pesos e constantes de cada camada são definidas pelas equações (3.13) e (3.14), respectivamente.

$$\frac{\partial C}{\partial \boldsymbol{w}^l} = \frac{\partial C}{\partial \boldsymbol{z}^l} \cdot \frac{\partial \boldsymbol{z}^l}{\partial \boldsymbol{w}^l} \tag{3.13}$$

$$\frac{\partial C}{\partial \boldsymbol{b}^l} = \frac{\partial C}{\partial \boldsymbol{z}^l} \cdot \frac{\partial \boldsymbol{z}^l}{\partial \boldsymbol{b}^l}. \tag{3.14}$$

O conjunto das derivadas parciais de $C(\boldsymbol{w}, \boldsymbol{b})$ em relação à $\boldsymbol{w}^l$ e $\boldsymbol{b}^l$ para todas as L camadas constituem o gradiente $\nabla C(\boldsymbol{w}, \boldsymbol{b})$ da equação (3.11).

3.1.2 Redes neurais convolucionais

O Aprendizado Profundo é composto por técnicas que utilizam redes neurais artificiais de diversas maneiras, possuindo sempre mais que duas camadas para que sejam classificadas como profundas (NIELSEN, 2015). Dentre elas, uma das mais utilizadas é a rede neural convolucional (CNN). Segundo LeCun et al. (1989), uma rede neural convolucional é um tipo de rede neural artificial especializada no processamento de dados que possuem uma topologia em forma de matriz conhecida. Dados de séries temporais, que podem ser pensadas como matrizes de uma dimensão com amostras tomadas em períodos regulares de tempo, e dados de imagens, que podem ser pensados como matrizes de pixels em duas dimensões, são

exemplos de estruturas em que as redes neurais convolucionais são especialistas. São ditas convolucionais por empregarem uma operação matemática denominada convolução, que é um tipo especial de operação linear. Define-se, assim, uma rede neural convolucional como sendo uma rede neural artificial que utiliza a operação de convolução no lugar de um produto matricial genérico em, pelo menos, uma de suas camadas (LECUN et al., 1989).

3.1.2.1 Camada de convolução

A operação de convolução, $S(t)$ entre duas funções, $y(t)$ e $h(t)$, é definida por:

$$v(t) = (y * h)(t) = \int_{-\infty}^{\infty} y(\tau)h(t-\tau)d\tau. \tag{3.15}$$

Na terminologia de redes neurais convolucionais, o primeiro argumento da convolução, $y(t)$, é chamada de entrada, enquanto o segundo argumento, $h(t)$, é chamado de *kernel*. A saída, $v(t)$, é algumas vezes referida como mapa de características (GOODFELLOW et al., 2016).

Quando os dados são trabalhados em um computador, o tempo é discretizado, e o sinal de entrada é fornecido em intervalos de tempo regulares e o índice de tempo t pode assumir somente valores inteiros, definindo assim a convolução discreta,

$$v(t) = (y * h)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau). \tag{3.16}$$

Em aplicações de Aprendizado de Máquina, a entrada normalmente é um conjunto de dados multidimensional, e o *kernel* é um conjunto de parâmetros multidimensional que são adaptados pelo algoritmo de aprendizagem. Como cada elemento da entrada e do *kernel* precisa ser explicitamente armazenado separadamente, é assumido que essas funções são nulas em todos os pontos exceto em um conjunto finito onde os valores são armazenados. Na prática, com isso é possível implementar uma soma infinita como sendo a soma do conjunto finito dos elementos armazenados (GOODFELLOW et al., 2016).

É possível utilizar as convoluções em mais de um eixo por vez. No processamento de imagens, é comum fazer o uso de convoluções e *kernels* em duas dimensões. Considerando uma imagem I de dimensão $i_I \times j_I$ como entrada, e um *kernel* K de dimensão $m_K \times n_K$, define-se a convolução discreta em duas dimensões,

$$S(i_I, j) = (I * K)(i_I, j_I) = \sum_{m_K} \sum_{n_K} I(m_K, n_K) K(i_I - m_K, j_I - n_K). \tag{3.17}$$

A operação de convolução é comutativa (GOODFELLOW et al., 2016), permitindo que a equação (3.17) possa ser escrita na forma

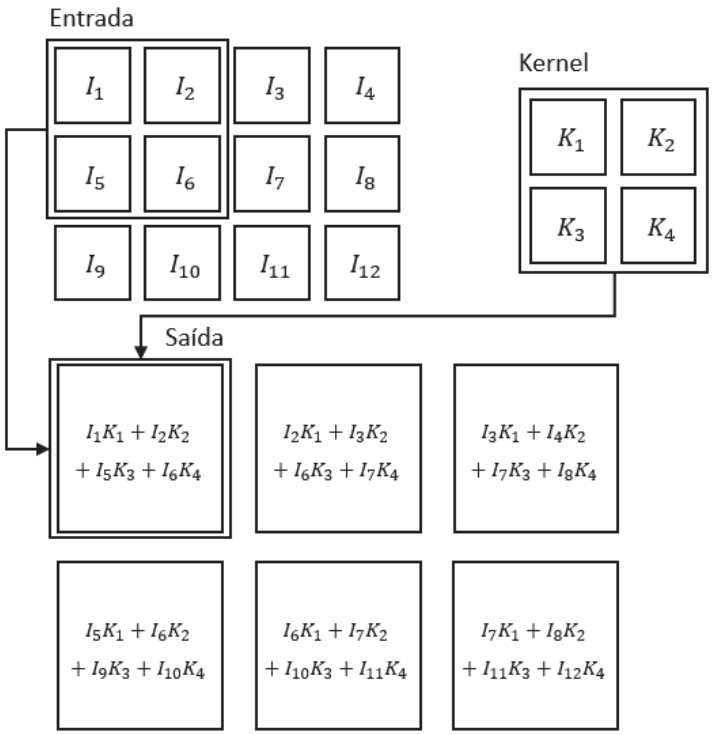
$$S(i_I, j_I) = (K * I)(i_I, j_I) = \sum_{m_K} \sum_{n_K} I(i_I - m_K, j_I - n_K) K(m_K, n_K). \tag{3.18}$$

A equação (3.18) possui uma implementação mais direta em bibliotecas de Aprendizado de Máquina, pois há uma variação menor no alcance dos valores válidos de m_Ke n_K. Essa propriedade comutativa surge devido à rotação do *kernel* relativo à entrada, no sentido de que, com o incremento de m, o índice na entrada também incrementa, mas o índice no *kernel* diminui (GOODFELLOW et al., 2016). Enquanto a propriedade comutativa é útil para validação, ela não se apresenta como uma propriedade importante para fins de implementação de uma rede neural convolucional. Muitas bibliotecas implementam uma função relacionada chamada de correlação cruzada (3.19), como é o caso da interface de programação de aplicação Keras (KERAS, 2019), que representa a mesma operação da convolução, mas sem rotacionar o *kernel*.

$$S(i_I, j_I) = (K * I)(i_I, j_I) = \sum_{m_K} \sum_{n_K} I(i_I + m_K, j_I + n_K) K(m_K, n_K). \tag{3.19}$$

A FIGURA 5 ilustra a operação representada pela equação (3.19), onde a saída é restringida somente em posições onde o *kernel* está totalmente contido na matriz de entrada, chamada de convolução válida. As caixas com setas indicam como o elemento superior-esquerdo da saída é formada quando o *kernel* é aplicado na região superior-esquerda correspondente da entrada.

FIGURA 5 – EXEMPLO DE CONVOLUÇÃO 2-D SEM ROTAÇÃO DE *KERNEL*



FONTE: Adaptado de GOODFELLOW et al. (2016).

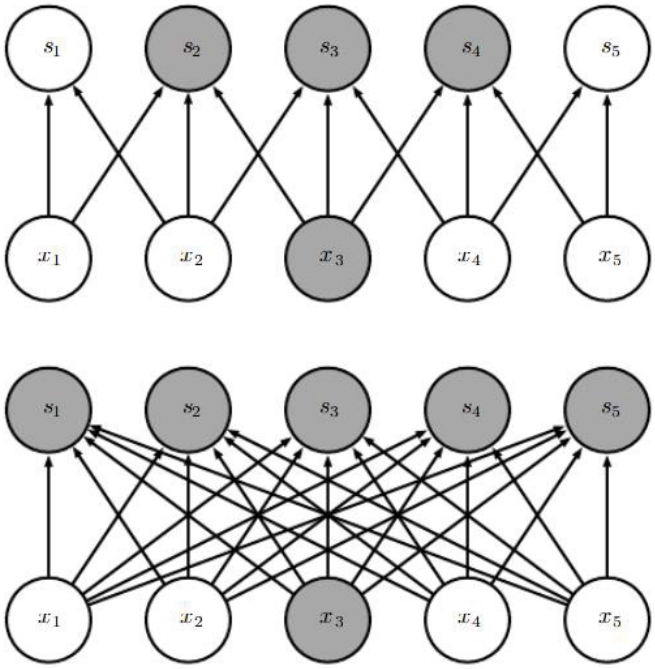
A convolução traz três conceitos importantes que podem melhorar um sistema de Aprendizado de Máquina: interações esparsas, compartilhamento de parâmetros e representações equivariantes.

Redes neurais artificiais tradicionais usam multiplicação de uma matriz por uma matriz de parâmetros com um parâmetro separado descrevendo a interação entre cada unidade de entrada e saída. Redes neurais convolucionais, entretanto, possuem interações esparsas (também chamadas de pesos esparsos), pois o *kernel* é menor que a entrada. Isso significa um armazenamento de menos parâmetros, que reduz a quantidade de memória necessária do modelo e melhora a eficiência estatística, além demandar um custo de processamento menor: se há x entradas e y saídas, a matriz de multiplicação precisa de $x \times y$ parâmetros, e os algoritmos utilizados na prática têm $O(x \times y)$ tempo de execução. Se o número de conexões que cada saída tem for limitado para k, então a aproximação esparsamente conectada necessita de $k \times y$ parâmetros e $O(k \times y)$ tempo de execução (GOODFELLOW et al., 2016).

A FIGURA 6 ilustra a diferença entre interações entre entrada e saída de uma conexão esparsa e uma conexão totalmente conectada. A entrada x_3 é destacada,

assim como as saídas s que são afetadas por ela, de acordo com a topologia. No diagrama superior, representando as interações esparsas, quando s é formado por uma convolução com largura de *kernel* igual a três, somente três saídas são afetadas pela entrada x. Quando s é formado por uma matriz de multiplicação, as conexões deixam de ser esparsas, então todas as saídas são afetadas por todas as entradas, como mostrado no diagrama inferior da FIGURA 6.

FIGURA 6 – INTERAÇÕES ESPARSAS



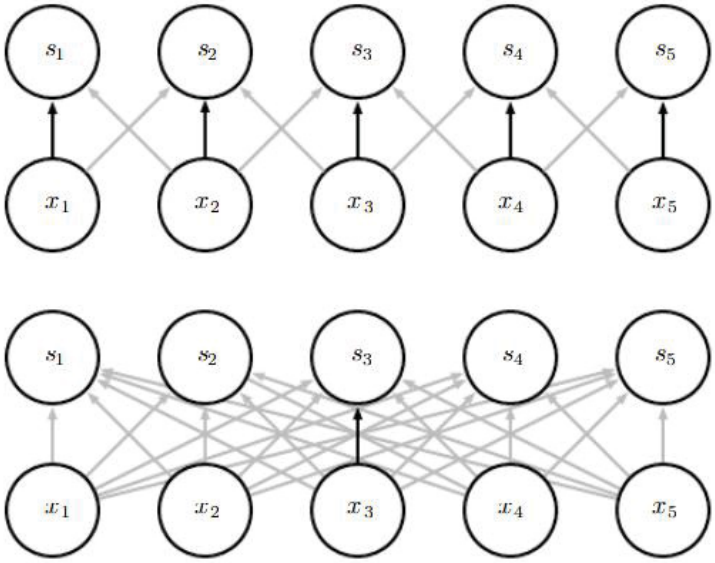
FONTE: Adaptado de GOODFELLOW et al. (2016).

O compartilhamento de parâmetros se refere ao uso dos mesmos parâmetros por mais de uma função no modelo. Em redes neurais artificiais tradicionais, cada elemento da matriz de pesos é usado exatamente uma vez para calcular a camada de saída, ou seja, é multiplicado por um elemento da entrada e não é mais utilizado. Em uma rede neural convolucional, cada membro do *kernel* é usado em cada posição da entrada. O compartilhamento de parâmetros na convolução significa que, ao invés de aprender um conjunto de parâmetros separados para cada região, somente um conjunto é aprendido. Isso não afeta o tempo de execução da alimentação direta da rede, mas reduz o armazenamento para k parâmetros (GOODFELLOW et al., 2016).

A FIGURA 7 apresenta uma comparação entre um modelo com compartilhamento de parâmetro e outro sem. No diagrama superior, as setas em

destaque indicam a utilização de um elemento central de uma rede neural convolucional com *kernel* de três elementos. Por causa do compartilhamento de parâmetros, esse parâmetro é usado por todas as saídas. No diagrama inferior, a seta em destaque mostra o uso de um elemento central de uma matriz de ganhos em um modelo totalmente conectado que, por não possuir compartilhamento de parâmetros, é utilizado por uma única saída.

FIGURA 7 – COMPARTILHAMENTO DE PARÂMETROS



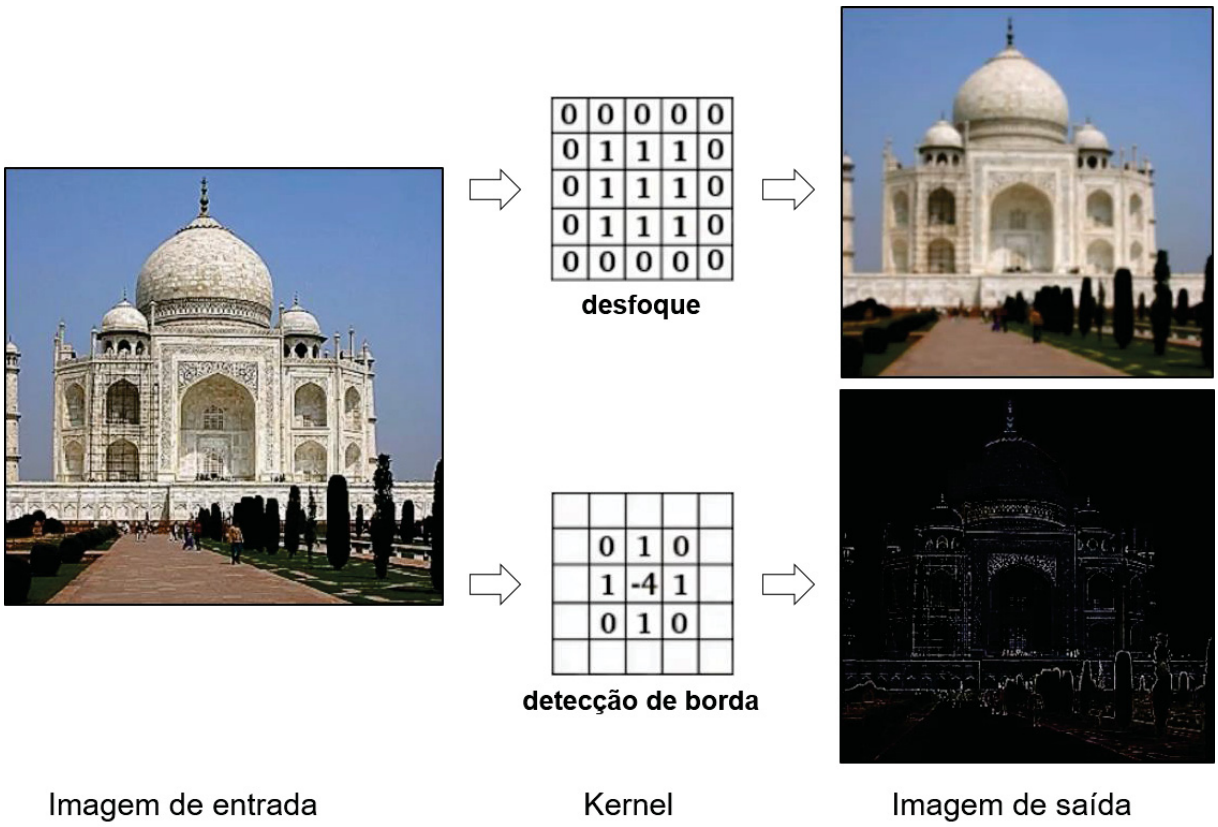
FONTE: Adaptado de GOODFELLOW et al. (2016).

No caso da convolução, a forma com que o compartilhamento de parâmetros ocorre concede para a camada a propriedade de equivariância à translação. Uma função $f(x)$ é equivariante a uma função $g(x)$ se $f\big(g(x)\big) = g\big(f(x)\big)$. Na convolução, se $g(x)$ for qualquer função que translada a entrada, então a função de convolução é dita equivariante a $g(x)$. Quando dados de séries temporais são processadas, isso significa que a convolução produz um tipo de linha do tempo que identifica quando propriedades diferentes aparecem na entrada. Se um evento é atrasado ou adiantado na entrada, a mesma representação desse evento aparecerá na saída, também atrasado ou adiantado. No caso de imagens, a convolução cria um mapa bidimensional de onde certas propriedades aparecem na entrada. Se o objeto da imagem na entrada for transladado, a representação na saída será transladada da mesma forma. A convolução não é naturalmente equivariante à algumas transformações, tais como mudanças de escala e rotação, necessitando de outros

mecanismos para poder manusear esses tipos de transformações (GOODFELLOW et al., 2016).

O *kernel* fornece, nesse caso, atua como um extrator de propriedades da entrada. Como exemplo, quando as amostras de entrada são imagens, o *kernel* atua como um filtro. A FIGURA 8 apresenta possíveis aplicações de *kernels* que atuam como filtros de imagem.

FIGURA 8 – EXEMPLOS DE *KERNEL* APLICADO COMO FILTRO DE IMAGEM



FONTE: Adaptado de EREMENKO (2018).

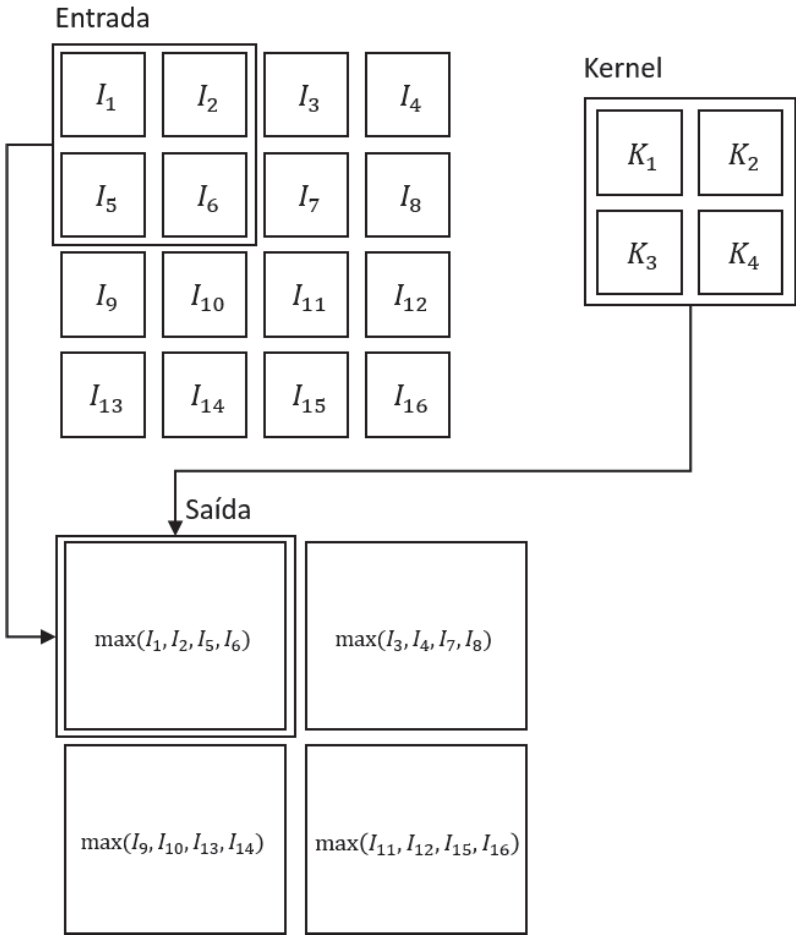
3.1.2.2 Camada de subamostragem

As camadas de convolução são normalmente seguidas por camadas de subamostragem com o objetivo de diminuir a resolução do mapa de propriedades e, consequentemente, a quantidade de parâmetros e o custo computacional. Operações de valor máximo e valor médio são dois métodos de subamostragem típicas na implementação de CNNs. Diferentemente dos *kernels* de convolução, os *kernels* de subamostragem permanecem inalterados durante o treinamento e a inferência. Outra

justificativa para a subamostragem é o aumento de robustez relacionado à distorções e erros causados por neurônios individualmente (ZHANG et al., 2019).

A subamostragem possui dois parâmetros essenciais: tamanho do *kernel* e o passo de varredura do *kernel*. Em alguns casos, essas duas variáveis podem se estender para até quatro parâmetros, onde o *kernel* não possui tamanho e varredura simétricos. A FIGURA 9 apresenta uma camada de subamostragem de valor máximo, onde um *kernel* simétrico, K, de tamanho 2, varre uma amostra de entrada, I, com uma varredura de passo simétrico, também de tamanho 2, mapeando a entrada na saída.

FIGURA 9 – EXEMPLO DE SUBAMOSTRAGEM DE VALOR MÁXIMO



FONTE: O autor (2019).

Alguns projetos não utilizam camadas de subamostragem quando o foco é aprender a posição específica de um objeto, como é o caso do compilador quântico desenvolvido no presente trabalho, onde a posição de cada porta quântica é fundamental e não pode ser subamostrada.

3.1.3 Aprendizado por reforço

Aprendizado por Reforço é um conjunto de técnicas de Aprendizado de Máquina que se localizam entre o grupo das técnicas de Aprendizado supervisionado e não supervisionado, caracterizados por possuírem ou não rótulos objetivos para cada exemplo, respectivamente. No Aprendizado por Reforço, existem rótulos esparsos e rótulos atrasados, denominados de recompensa, nos quais o agente se baseia exclusivamente para aprender e se comportar no ambiente.

Aplicações de Aprendizado por Reforço abrangem diversas áreas, tais como veículos autônomos, robôs e jogos eletrônicos. Ng et al. (2004) treinaram um controlador de um helicóptero real, utilizando Aprendizado por Reforço, para realizar manobras acrobáticas. O estado do helicóptero foi descrito por um vetor de doze dimensões e seu espaço de ações possui seis dimensões. Levine e Koltun (2014) treinaram um robô bípede simulado para executar uma tarefa de caminhada em terrenos regulares e irregulares. Eles também testaram a habilidade do robô de se recuperar após um empurrão. Para isso, foi utilizada uma variante de um regulador quadrático linear, chamado de regulador quadrático linear iterativo, com o objetivo de minimizar a entropia relativa entre uma trajetória ótima e uma trajetória criada pelo modelo de Aprendizado por Reforço. Silver et al. (2016) desenvolveram um algoritmo chamado AlphaGo, que foi capaz de aprender a jogar o milenar jogo chinês Go, utilizando uma *Deep Q-Network* e realizando partidas consigo mesmo. Em 2016, esse algoritmo foi capaz de derrotar o dezoito vezes campeão mundial, Lee Sedol.

Os itens a seguir apresentam as técnicas que compõe a base para o *Q-learning*, bem como a sua integração com técnicas de Aprendizado Profundo para compor o chamado *Deep Q-Learning*.

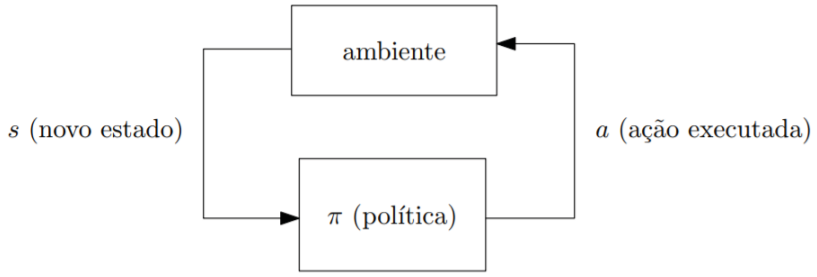
3.1.3.1 Processo de decisão de Markov

Segundo Puterman (1994), o processo de decisão de Markov (MDP) é uma forma de modelar processos onde as transições entre estados são probabilísticas. Esse tipo de modelagem permite a visualização do estado em que o processo está e a interferência do processo periodicamente, chamada de ação, que resulta em uma recompensa dependende do estado onde o processo se encontra. São ditos de Markov (ou Markovianos) porque os processos modelados obedecem a propriedade

de Markov: o efeito de uma ação em um estado depende apenas da ação e do estado atual do sistema (e não de como o processo chegou a tal estado). São chamados de processos de decisão porque modelam a possibilidade de um agente (ou tomador de decisões) interferir periodicamente no sistema executando ações, diferentemente das cadeias de Markov, onde não se trata de como interferir no processo (PELLEGRINI; WAINER, 2007).

O processo se baseia em um agente, situado em um ambiente, que se encontra em um determinado estado. O agente realiza uma ação no ambiente que pode ou não resultar em uma recompensa. Essas ações transformam o ambiente e levam à um novo estado onde o agente pode realizar uma outra ação, e assim por diante. As regras que o agente usa para tomar uma ação são chamadas de política. O ambiente normalmente é estocástico, o que significa que o próximo estado pode ser aleatório (PELLEGRINI; WAINER, 2007).

FIGURA 10 – FUNCIONAMENTO DE UM SISTEMA MODELADO COMO MDP



FONTE: Adaptado de PELLEGRINI e WAINER (2007).

O conjunto de estados e ações, junto com as regras de transição de um estado para outro e de ganho de recompensa formam um processo de decisão de Markov. Um episódio desse processo forma uma sequência finita de estados (s), ações (a) e recompensas (r) s_0, a_0, r_1, s_1, a_1, r_2, s_2, ..., s_{n-1}, a_{n-1}, r_n, s_n. Para cada ação a_i realizada no estado s_i, uma recompensa r_{i+1} é atribuída. O episódio termina quando o agente se encontra no estado final s_n.

3.1.3.2 Recompensa futura descontada

Para ser bem-sucedido a longo prazo, é necessário levar em consideração não apenas as recompensas imediatas, mas também as recompensas

futuras que o agente irá obter. Dado uma execução no processo de decisão de Markov, a recompensa total de um episódio é dada por:

$$R = r_1 + r_2 + r_3 + \cdots + r_n. \tag{3.20}$$

Dado isso, a recompensa futura total, a partir do ponto t pode ser expressada como:

$$R_t = r_t + r_{t+1} + r_{t+2} + \cdots + r_n. \tag{3.21}$$

Devido ao ambiente ser estocástico, não é possível prever se o agente irá obter a mesma recompensa na próxima vez que efetuar as mesmas ações. Quanto maior for o período futuro considerado, maior poderá ser a divergência. Por esta razão, é comum a utilização da chamada recompensa futura descontada, definida como

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n, \tag{3.22}$$

onde γ é o fator de desconto, definido entre 0 e 1. Quanto mais distante no futuro for a recompensa considerada, menos ela será considerada. Pode-se demonstrar também que a recompensa futura descontada no ponto t pode ser expressada nos mesmos termos do ponto $t + 1$, tal que

$$R_t = r_t + \gamma[r_{t+1} + \gamma(r_{t+2} + \cdots)] = r_t + \gamma R_{t+1}. \tag{3.23}$$

Se o fator de desconto for definido como nulo, a estratégia definida será de curto prazo e as ações serão regidas somente pelas recompensas imediatas. Se o ambiente for determinístico e as mesmas ações resultarem sempre nas mesmas recompensas, então pode-se definir o fator de desconto igual à unidade, considerando assim o mesmo peso para todas as recompensas, imediatas e futuras. Uma estratégia válida será sempre escolher a ação que maximiza a recompensa futura descontada.

3.1.3.3 *Q-Learning*

Segundo Watkins (1989), *Q-learning* é uma forma de Aprendizado por Reforço que não necessita de um modelo. Essa técnica fornece agentes com a capacidade de aprender e realizar ações otimizadas em um domínio Markoviano, por meio da experiência obtida como consequência das ações tomadas, sem necessitar da construção de mapas dos domínios. Neste método, uma função $Q(s,a)$ é definida como a representação da recompensa futura descontada quando uma ação a no estado s é efetuada e continua otimizada a partir do ponto t, definida como

$$Q(s_t, a_t) = \max_{\pi} R_{t+1}. \tag{3.24}$$

A função $Q(s,a)$ também pode ser definida como a melhor recompensa possível, no final de um episódio, quando a ação a é executada no estado s. Tal função é assim chamada pois representa a qualidade de uma certa ação em um dado estado. Considerando a existência da função $Q(s,a)$, a política π de decisão de uma ação será a que resultará na maior recompensa final, ou seja, escolha a ação que resulta no maior valor de Q,

$$\pi(s) = \operatorname*{argmax}_{a} Q(s,a). \tag{3.25}$$

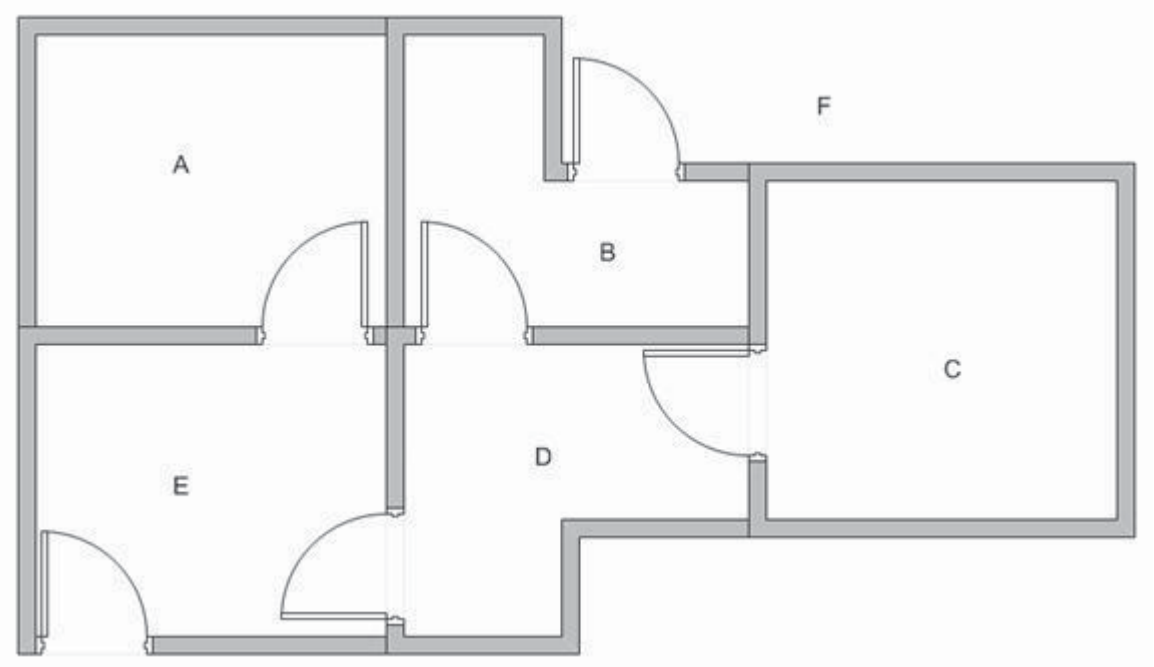
A função $Q(s,a)$ é obtida de forma semelhante à recompensa futura descontada, pela equação de Bellman (3.26), que define a máxima recompensa futura para um dado estado e uma dada ação como sendo a soma da recompensa imediata com a máxima recompensa futura possível no próximo estado, tal que

$$Q(s_t, a_t) = r + \gamma \max_{a} Q(s_{t+1}, a_{t+1}). \tag{3.26}$$

Define-se, assim, *Q-learning* como sendo o processo iterativo de aproximação da função $Q(s,a)$ real, por meio da equação de Bellman.

Para fins de exemplificação, suponha-se uma construção contendo cinco cômodos, conectados por meio de portas, conforme representado na FIGURA 11.

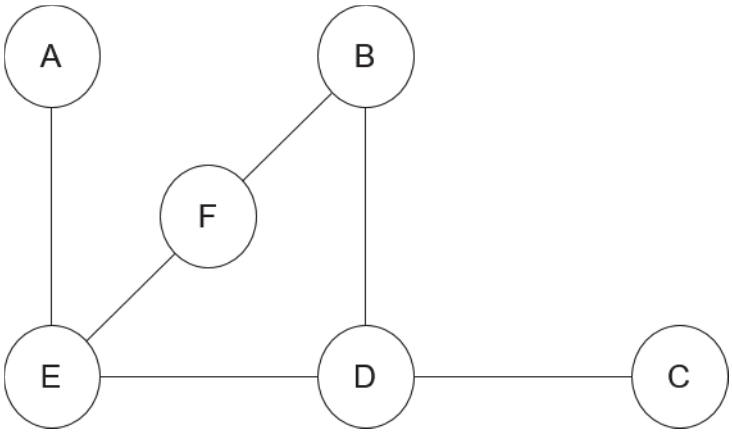
FIGURA 11 – PROBLEMA DE EXEMPLIFICAÇÃO – *Q-LEARNING*



FONTE: Adaptado de TEKNOMO (2019).

Cada cômodo é identificado pelas letras sequenciais de A à E, e a parte externa à construção é identificada pela letra F, que pode ser acessada pelas portas externas nos cômodos B e E. Os cômodos e as portas podem ser representados em forma de grafos, por meio de transições (portas) e estados (cômodos), como ilustrado na FIGURA 12.

FIGURA 12 – GRAFO DE EXEMPLIFICAÇÃO – *Q-LEARNING*

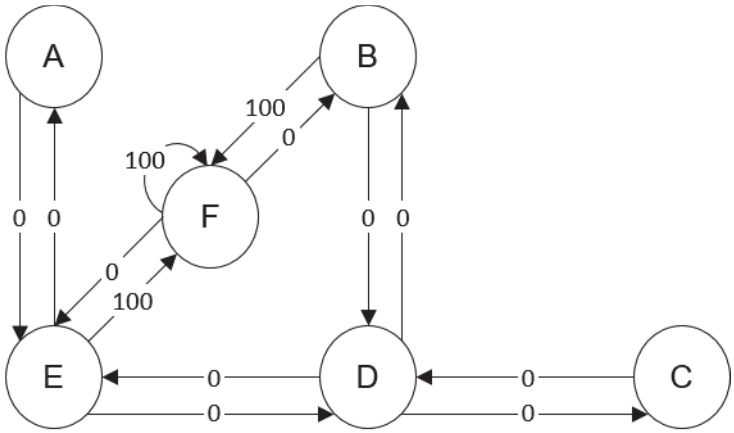


FONTE: Adaptado de TEKNOMO (2019).

Como parte do exemplo, o estado F é definido como objetivo de um agente colocado em qualquer um dos outros estados, ilustrando que, dado um agente

localizado em um determinado cômodo da construção, o objetivo do aprendizado é encontrar o menor caminho para fora dessa construção. Para isso, são definidas as recompensas instantâneas para cada transição entre os estados, sendo definido o valor 100 para as transições (portas) que levam para o estado F (parte externa à construção) e 0 para as restantes. Como o agente pode retornar para estados anteriores, são definidas duas recompensas para cada transição, uma para cada sentido. Uma transição adicional em forma de laço, com recompensa instantânea igual a 100, é necessária para que o agente não transite para outros estados uma vez que alcance o objetivo. Define-se, assim, o processo de decisão de Markov para este exemplo, apresentado na FIGURA 13.

FIGURA 13 – PROCESSO DE DECISÃO DE MARKOV DO EXEMPLO – *Q-LEARNING*



FONTE: Adaptado de TEKNOMO (2019).

Supõe-se que o agente se encontra do estado C. A partir desse estado, o agente pode ir somente para o estado D, mas, a partir desse, é possível ir tanto para o estado B quanto para o estado E, bem como retornar ao estado C. A partir de todas as ações, estados e recompensas do processo, define-se a função de recompensas imediatas, $r(s, a)$, por meio da TABELA 1.

TABELA 1 – RECOMPENSAS IMEDIATAS DO EXEMPLO – *Q-LEARNING*

		posição futura do agente (ação)					
		A	**B**	**C**	**D**	**E**	**F**
posição atual do agente (estado)	**A**	-	-	-	-	0	-
	B	-	-	-	0	-	100
	C	-	-	-	0	-	-
	D	-	0	0	-	0	-
	E	0	-	-	0	-	100
	F	-	0	-	-	0	100

Fonte: O autor (2019).

As células referenciadas com um hífen são ditas ações impossíveis, pois não estão presentes no diagrama do processo de decisão de Markov definido na FIGURA 13.

Com a função de recompensas imediatas definida, uma segunda função, $Q(\mathrm{s},\mathrm{a})$, é declarada, inicialmente com todos os valores nulos. Essa função é atualizada, iterativamente, pela função de Bellman, $Q(s_t, a_t)$, definida pela equação (3.26). A função $Q(\mathrm{s},\mathrm{a})$, na sua condição inicial, é apresentada pela TABELA 2.

TABELA 2 – ESTADO INICIAL DA FUNÇÃO Q DO EXEMPLO – *Q-LEARNING*

		posição futura do agente (ação)					
		A	**B**	**C**	**D**	**E**	**F**
posição atual do agente (estado)	**A**	0	0	0	0	0	0
	B	0	0	0	0	0	0
	C	0	0	0	0	0	0
	D	0	0	0	0	0	0
	E	0	0	0	0	0	0
	F	0	0	0	0	0	0

Fonte: O autor (2019).

Ainda para fins de exemplo, atribui-se o fator de desconto, γ, como 0,8, e o estado inicial do agente como B. Analisando a segunda linha da matriz de recompensas, $r(s,a)$, da TABELA 1, há duas possibilidades de ações para o atual estado, B, que é ir para o estado D ou para o estado F. Por meio de seleção arbitrária, supõe-se a ação do agente de ir para o estado F. A partir desse estado, é possível

transitar para os estados B, E e F. Aplicando-se a função de Bellman para a transição de B para F, e considerando as possibilidades de transições futuras, a função $Q(\mathrm{s},\mathrm{a})$ é atualizada pela função de Bellman da seguinte forma:

$$Q(\mathrm{B},\mathrm{F}) = r(B,F) + 0{,}8 \max_{a}\{Q(\mathrm{F},\mathrm{B}), Q(\mathrm{F},\mathrm{E}), Q(\mathrm{F},\mathrm{F})\} = 100 + 0{,}8 \cdot 0 = 100. \tag{3.27}$$

Como a função $Q(\mathrm{s},\mathrm{a})$ possui, inicialmente, todos os valores nulos, $Q(\mathrm{F},\mathrm{B})$, $Q(\mathrm{F},\mathrm{E})$ e $Q(\mathrm{F},\mathrm{F})$ são nulos, fazendo com que o termo $\max_{a} Q(s_{t+1}, a_{t+1})$ seja igual a zero. A função $Q(\mathrm{s},\mathrm{a})$, inicialmente definida pela TABELA 2, agora é atualizada para a TABELA 3.

TABELA 3 – ESTADO SECUNDÁRIO DA FUNÇÃO Q DO EXEMPLO – *Q-LEARNING*

		posição futura do agente (ação)					
		A	**B**	**C**	**D**	**E**	**F**
posição atual do agente (estado)	**A**	0	0	0	0	0	0
	B	0	0	0	0	0	100
	C	0	0	0	0	0	0
	D	0	0	0	0	0	0
	E	0	0	0	0	0	0
	F	0	0	0	0	0	0

Fonte: O autor (2019).

O próximo estado, F, se torna o estado atual. Como F é o estado objetivo, um episódio é finalizado. Seguindo este algoritmo por meio de vários episódios, a função $Q(\mathrm{s},\mathrm{a})$ converge e, normalizando o valor de recompensa máximo em 100, resulta na TABELA 4.

TABELA 4 – ESTADO FINAL DA FUNÇÃO Q DO EXEMPLO – *Q-LEARNING*

		posição futura do agente (ação)					
		A	**B**	**C**	**D**	**E**	**F**
posição atual do agente (estado)	**A**	0	0	0	0	80	0
	B	0	0	0	64	0	100
	C	0	0	0	64	0	0
	D	0	80	51	0	80	0
	E	64	0	0	64	0	100
	F	0	80	0	0	80	100

Fonte: O autor (2019).

Por meio da equação (3.25), a solução que possui a maior recompensa final é encontrada, ou seja, para o exemplo apresentado, o menor caminho que leva o agente de um cômodo inicial para o ambiente externo à construção. No caso apresentado, se o agente iniciar no estado C, há dois caminhos que geram a mesma recompensa final: C – D – B – F ou C – D – E – F, ambos resultando em 244 pontos de recompensa e escolhidos arbitrariamente, possibilitando mais de uma resposta para o mesmo problema.

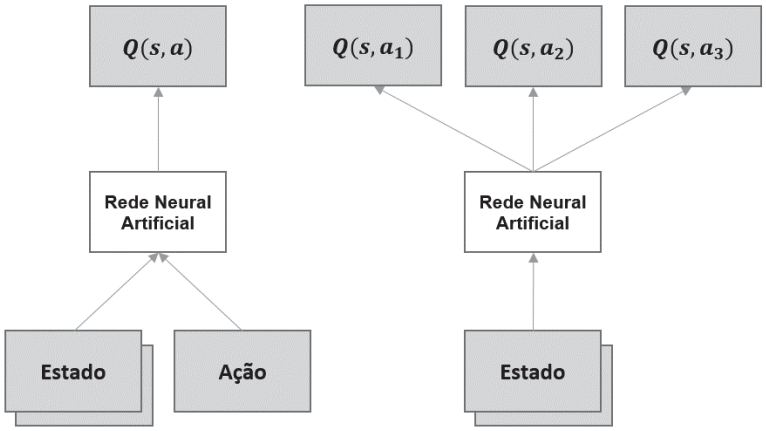
3.1.3.4 *Deep Q-Learning*

Segundo Mnih et al. (2015), o estado de um ambiente que representa um jogo, por exemplo, pode ser definido genericamente pelo arranjo de pixels na tela que, implicitamente, contém toda a informação relevante sobre a situação do jogo, exceto sobre a movimentação, que pode ser obtida utilizando duas telas consecutivas. Considerando uma tela de um jogo de 84x84 pixels, em escala de cinza de 8 bits (256 intensidades), que utiliza 4 imagens em sequência para obter informações de movimentação, seriam necessários $256^{84\times84\times4} \approx 10^{67970}$ estados para representar todas as possibilidades desse jogo. Como a frequência de cada estado visitado pelo agente não é uniforme, a raridade de alguns eventos faria com que o tempo para que o processo iterativo de aproximação da função $Q(s,a)$ convergisse fosse extremamente elevado. Além disso, estados nunca visitados durante o processo de aprendizado não teriam um valor coerente na função $Q(s,a)$ (MNIH et al., 2015).

Os avanços recentes em visão computacional e reconhecimento de voz estão baseados em treinamentos eficientes de redes neurais artificiais profundas utilizando grandes conjuntos de treinamentos. Alimentando dados suficientes, a rede normalmente consegue aprender representações melhores que extratores feitos a mão. Com isso, é possível representar a função $Q(s,a)$ por meio de uma rede neural artificial que utiliza o estado atual e as ações do agente como entrada crua, e a saída como valor de $Q(s,a)$. Também é possível utilizar somente o estado atual como entrada e obter um valor de $Q(s,a)$ para cada possível ação. Essa aproximação possui a vantagem de demandar apenas uma alimentação direta para obter todos os valores de $Q(s,a)$ (MNIH et al., 2015).

A FIGURA 14 apresenta duas topologias possíveis de representação da função $Q(s,a)$ utilizando redes neurais artificiais. A topologia à esquerda demanda uma ação como entrada e gera apenas um valor de $Q(s,a)$ para cada alimentação direta. A topologia à direita, por sua vez, toma como entrada somente o estado atual, e gera os valores de $Q(s,a)$ para todas as ações possíveis.

FIGURA 14 – TOPOLOGIAS DE *DEEP Q-NETWORK*



FONTE: Adaptado de MNIH et al. (2015).

Os valores de $Q(s,a)$ podem ser quaisquer valores reais, fazendo com que a rede tenha uma tarefa de regressão, que pode ser efetuada minimizando uma função custo de erro médio quadrático, tal que

$$L = \frac{1}{2}\left[r + \gamma \max_{a} Q(s_{t+1}, a_{t+1}) - Q(s,a)\right]^2, \tag{3.28}$$

onde a parte à esquerda do sinal de subtração representa a saída esperada e a parte à direita do sinal de subtração representa a predição.

Assim sendo, dada uma transição conhecida $\langle s_t, a_t, r_t, s_{t+1} \rangle$ e o fator de desconto γ, a regra de atualização da função $Q(s, a)$ é a seguinte (MNIH et al., 2015):

1. Alimentar a rede diretamente com o estado atual s_t a fim de obter a predição de $Q(s_t, a_{i,t})$ para o conjunto de todas as possíveis ações $a_{i,t}$.
2. Alimentar a rede diretamente para o estado seguinte s_{t+1} a fim de obter a predição de $Q(s_{t+1}, a_{i,t+1})$ para o conjunto de todas as possíveis ações futuras $a_{i,t+1}$.
3. Aplicar a equação (3.28) utilizando o conjunto de $Q(s_{t+1}, a_{i,t+1})$ obtidos no passo 2 como $Q(s_{t+1}, a_{t+1})$ e $Q(s_t, a_t)$ como $Q(s, a)$.
4. Definir $L = 0$ na equação (3.28) para todas as outras ações $a_{i,t}$ diferentes de a_t.
5. Atualizar os pesos da rede neural artificial por meio da retropropagação do erro.

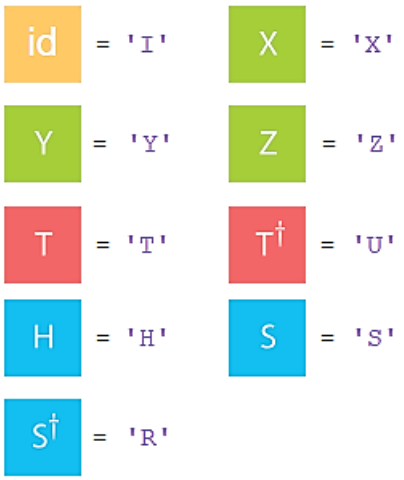
4 PREPARAÇÃO DE DADOS E AJUSTES DAS TÉCNICAS

Esta dissertação restringe-se ao computador quântico IBM-Q 5 Tenerife. As técnicas de Aprendizado de Máquina utilizarão os dados gerados por meio de um simulador desse dispositivo, validado com o simulador IBM-Q 5 Tenerife disponibilizado online pela IBM (IBM, 2019).

4.1 SIMULADOR DE DISPOSITIVOS QUÂNTICOS

Dentre os objetivos específicos definidos dessa dissertação está o desenvolvimento de um simulador de circuitos quânticos, baseado no IBM-Q 5 Tenerife (IBM, 2019), que é um processador quântico de 5 bits. Utilizando o ambiente computacional Matlab® 2018a foi implementado uma função que recebe uma *string,* onde cada linha representa um q-bit e cada coluna representa uma operação com cada q-bit. Foi adicionado um operador linear adicional para a representação de um espaço vazio, onde não é realizada nenhuma operação, sendo este operador uma matriz identidade. Cada porta quântica é, então, definida por um caractere no simulador desenvolvido. A FIGURA 15 apresenta a correlação entre as portas quânticas disponíveis de um q-bit no IBM-Q 5 Tenerife e o caractere que as representam no simulador desenvolvido.

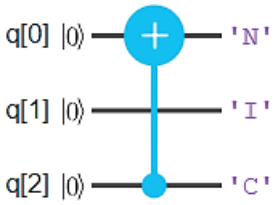
FIGURA 15 – REPRESENTAÇÃO DAS PORTAS QUÂNTICAS DE 1 Q-BIT



FONTE: O autor (2019).

Para a representação da porta CNOT quântica foram utilizados dois caracteres, pois essa porta opera em dois q-bits simultaneamente, sendo um de controle e um controlado. A FIGURA 16 mostra a representação da porta CNOT quântica em um sistema de 3 q-bits, onde o q-bit menos significativo $q[0]$ é o controlado e o q-bit mais significativo $q[2]$ é o controle. O q-bit $q[1]$ não passa por nenhuma mudança de estado durante esta operação, por isso ele recebe o operador linear identidade.

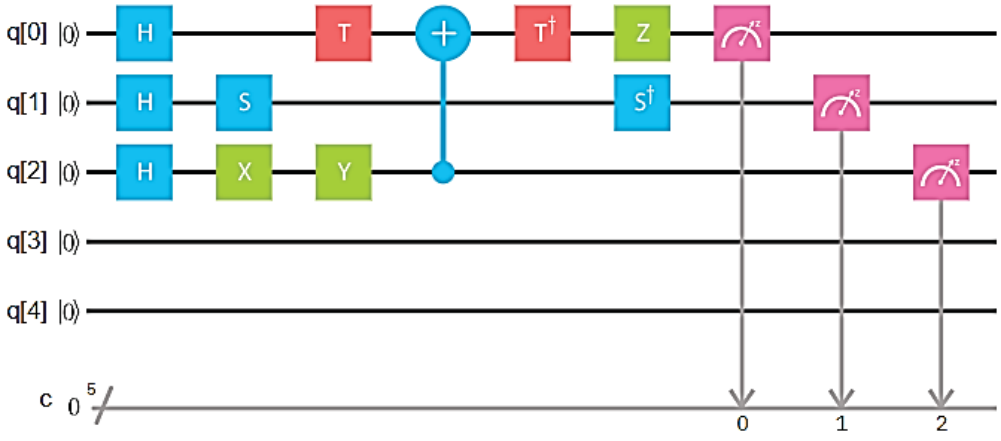
FIGURA 16 – REPRESENTAÇÃO DA PORTA CNOT QUÂNTICA



FONTE: O autor (2019).

Assim sendo, a FIGURA 17 apresenta um exemplo de operação equivalente entre o IBM-Q 5 Tenerife e o simulador desenvolvido, representado pela função em Matlab® na parte inferior da figura. Pode-se notar que os espaços vazios da grade são preenchidos com o operador linear identidade no simulador desenvolvido.

FIGURA 17 – REPRESENTAÇÃO EQUIVALENTE ENTRE CIRCUITOS

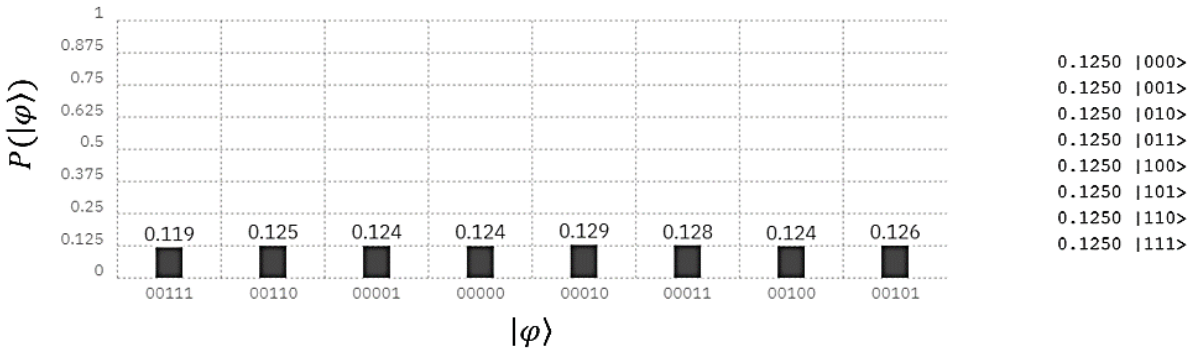



```
QuantumCircuit(['HITNUZ'; ...
                'HSIIIR'; ...
                'HXYCII'])
```

FONTE: O autor (2019).

A FIGURA 18 apresenta uma comparação entre os resultados apresentados pelo IBM-Q (à esquerda) e o simulador desenvolvido (à direita). Esse resultado é gerado pelo circuito apresentado na FIGURA 17. A discrepância entre os valores se dá porque o simulador desenvolvido considera portas quânticas ideais. O eixo das abcissas do gráfico à esquerda representa todos os possíveis estados de 3 q-bits, enquanto o eixo das ordenadas representa a probabilidade desses estados serem observados na saída do circuito. O mesmo resultado, apresentado de forma diferente, é apresentado à direita, na FIGURA 18, onde os valores representados na notação de Dirac são os possíveis estados para 3 q-bits, e os valores adjacentes são as probabilidades de cada estado ser observado na saída do circuito.

FIGURA 18 – APRESENTAÇÃO DOS RESULTADOS: IBM-Q E SIMULADOR DESENVOLVIDO

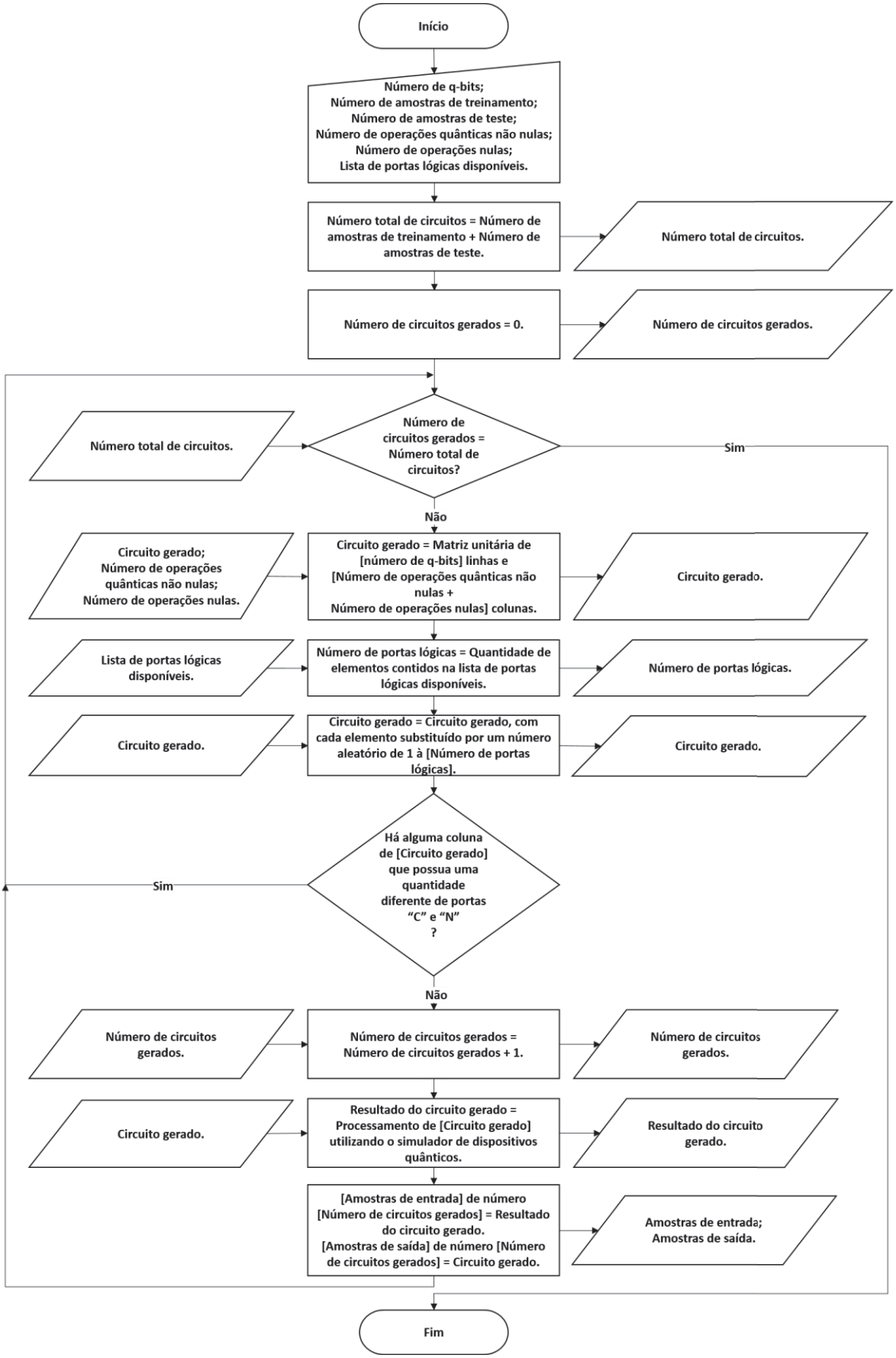


FONTE: O autor (2019).

4.2 GERADOR DE CONJUNTO DE DADOS

O simulador de dispositivos quânticos implementado viabilizou a automatização da criação de conjuntos de treinamentos e testes diversos, para qualquer quantidade de q-bits e operações quânticas. Para isso, é necessário fornecer a quantidade de circuitos de treinamento e de teste desejada, a quantidade de q-bits e o número máximo de operações. Com essas informações, o algoritmo gera um conjunto de circuitos aleatórios, mas que são validados a cada operação. Circuitos que possuem operações não permitidas, como ter um controle sem um q-bit controlado no caso da CNOT, são descartados, e a contagem de amostras só é incrementada quando a validação é confirmada. A FIGURA 19 apresenta o fluxograma do gerador de conjunto de dados, explicando passo a passo como os conjuntos de treinamento e teste são gerados.

FIGURA 19 – FLUXOGRAMA DO GERADOR DE CONJUNTO DE DADOS



FONTE: O autor (2019).

Em resumo, o algoritmo descrito pelo fluxograma da FIGURA 19 possui como argumentos o número de q-bits, o número de operações quânticas, o número de amostras do conjunto de treinamento e de testes e a lista de portas quânticas disponíveis no dispositivo quântico em que o compilador será utilizado (no caso dessa dissertação, as portas quânticas são as disponíveis no IBM-Q 5 Tenerife). O algoritmo inicia uma matriz unitária com o número de linhas e colunas igual ao número de q-bits e operações quânticas, respectivamente. Em seguida, uma varredura é feita em cada elemento dessa matriz, que é substituído por um número aleatório entre 1 e a quantidade de portas quânticas disponíveis no dispositivo quântico em que o compilador será utilizado, onde cada um desses valores representa uma porta quântica. Se essa matriz possuir algum de seus elementos que represente uma operação CNOT, o algoritmo verifica se o complemento da operação está presente na mesma coluna. Por exemplo, se a primeira coluna da matriz possuir o número que representa a operação de inversão da porta quânticas CNOT, mas não possuir o número que representa o controle da operação, o circuito é invalidado e não é adicionado no conjunto de dados. O conjunto de dados é, então, salvo em um arquivo que possui um conjunto de treinamento e um de testes que, posteriormente, é importado em Python para treinamento e validação das técnicas utilizando TensorFlow®.

4.3 PREPARAÇÃO DOS DADOS PARA ANN E CNN

Por meio dos conjuntos de treinamento e teste gerados pelo gerador de conjunto de dados desenvolvido, os dados são pré-processados antes de serem utilizados nas técnicas definidas no escopo deste trabalho, sendo eles diferentemente tratados em cada caso. Em comum, todos os circuitos dos conjuntos de dados de saída são convertidos de *strings* para números por meio de um mapeamento, definido pelo usuário de acordo com a disponibilidade de portas quânticas no dispositivo quântico que se deseja simular. No caso do IBM-Q 5 Tenerife, existem dez operações possíveis, sendo que uma delas, a CNOT, utiliza duas operações, uma para o q-bit de controle e uma para o q-bit controlado. Para os testes desenvolvidos neste trabalho, o mapeamento utilizado foi o apresentado na TABELA 5.

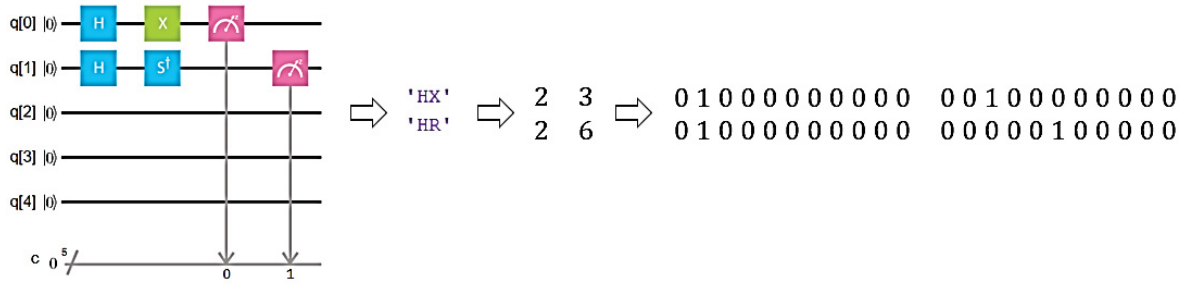
TABELA 5 – MAPEAMENTO DAS PORTAS QUÂNTICAS

porta quântica	valor relacionado
I	1
H	2
X	3
Y	4
Z	5
T	6
U	7
R	8
S	9
C	10
N	11

Fonte: O autor (2019).

A partir desse mapeamento, as técnicas de Aprendizagem adotadas se utilizam de pré-processamentos diferentes. Para a ANN multicamadas e a CNN, os circuitos mapeados são convertidos em conjuntos categóricos, formando uma matriz que possui uma quantidade de linhas igual à quantidade de amostras e uma quantidade de colunas igual ao produto da quantidade de espaços disponíveis no *grid* do circuito e a quantidade de valores contidos no mapeamento. Para se gerar uma amostra, o circuito é mapeado de acordo com a TABELA 5, e cada valor é convertido em uma sequência de valores binários com uma quantidade de valores igual à quantidade de portas quânticas disponíveis, onde todos os valores são nulos exceto pela posição que possui índice igual à da porta quântica que se deseja converter, a qual é atribuído o valor unitário. A FIGURA 20 apresenta um exemplo de conversão de um circuito de 2 q-bits e duas operações. O processo, da esquerda para a direita, converte o circuito modelado no IBM-Q 5 Tenerife em uma matriz de caracteres, utilizado como entrada do simulador de dispositivos quânticos implementado. Essa matriz é mapeada e, por fim, cada elemento é convertido no seu equivalente categórico.

FIGURA 20 – PRÉ-PROCESSAMENTO DOS DADOS DE SAÍDA (ANN E CNN)



FONTE: O autor (2019).

Sendo o circuito quântico a saída desejada dos modelos, dado um algoritmo quântico como entrada, e considerando a camada de saída das redes neurais como camadas totalmente conectadas, tanto para a ANN quanto para a CNN, é necessário que a matriz de saída do processo da FIGURA 20 seja convertida em uma sequência. Nesse caso, foi adotada a concatenação de linhas ao invés de colunas, simplesmente por requerer menos operações de conversões de formato nos *scripts* desenvolvidos. A FIGURA 21 apresenta um exemplo de conversão de matriz de elementos categóricos de um circuito quântico já mapeado em uma sequência concatenada.

FIGURA 21 – CONVERSÃO DA MATRIZ CATEGÓRICA EM UMA SEQUÊNCIA

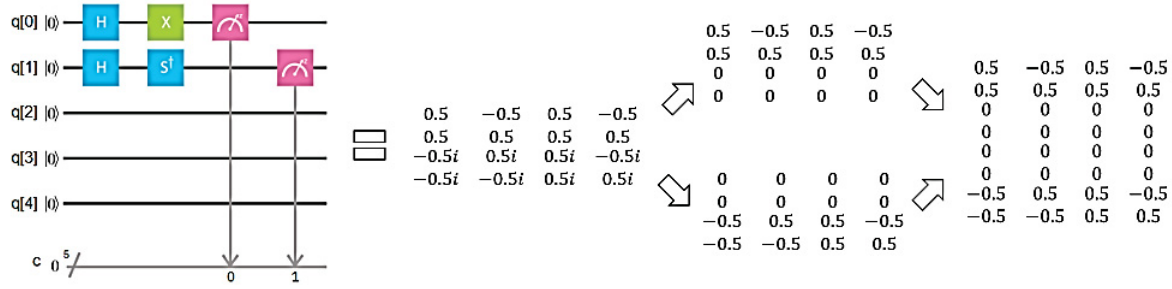
FONTE: O autor (2019).

Tanto para a ANN quanto para a CNN, essas sequências representam os valores de saídas esperados, ou seja, o circuito que gera o algoritmo dado como entrada. Assim sendo, a quantidade de neurônios na última camada para a ANN e a CNN é definida pela quantidade de elementos contida na sequência. Por exemplo, no caso do circuito com duas operações quânticas e dois q-bits apresentado, que resultam em quatro posições no *grid* a serem preenchidas com portas quânticas, e utilizando um dispositivo quântico com onze portas quânticas implementadas, como é o caso do IBM-Q 5 Tenerife, a quantidade de neurônios na camada de saída, tanto para a ANN quanto para a CNN, seria quarenta e quatro, sendo um conjunto de onze possíveis portas quânticas para cada uma das quatro posições no *grid* do circuito.

Os dados de entrada também possuem a forma de uma matriz, originalmente, já que representam a saída de um determinado circuito quântico. No caso da CNN,

esses dados de entrada não precisam ser convertidos em sequências, já que a técnica trabalha com dados nesse formato. Como são números complexos, as partes real e imaginária são separadas em duas matrizes e concatenadas em forma de coluna, com a matriz real seguida da matriz imaginária, definindo uma amostra de entrada. A FIGURA 22 apresenta a conversão da saída de um circuito quântico em uma amostra de entrada para a CNN.

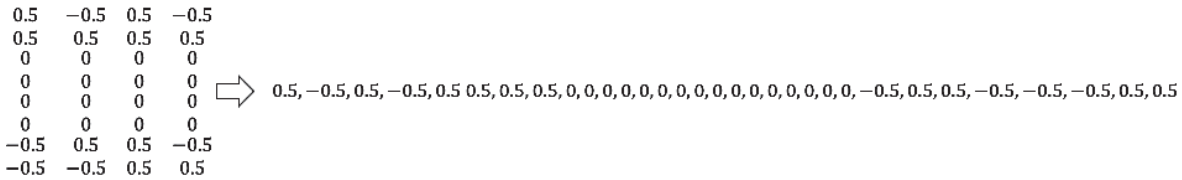
FIGURA 22 – PRÉ-PROCESSAMENTO DOS DADOS DE ENTRADA (CNN)



FONTE: O autor (2019).

No caso da ANN, é necessário realizar uma operação de conversão de formato adicional, de matriz para uma sequência, assim como é feito para os dados de saída. Da mesma forma, foi adotada a concatenação de linhas ao invés de colunas, para a reutilização de algoritmos e otimização de código. A FIGURA 23 apresenta a conversão adicional para os dados de entrada da ANN.

FIGURA 23 – PRÉ-PROCESSAMENTO DOS DADOS DE ENTRADA (ANN)



FONTE: O autor (2019).

4.4 AMBIENTE DE INTERAÇÃO PARA DQN

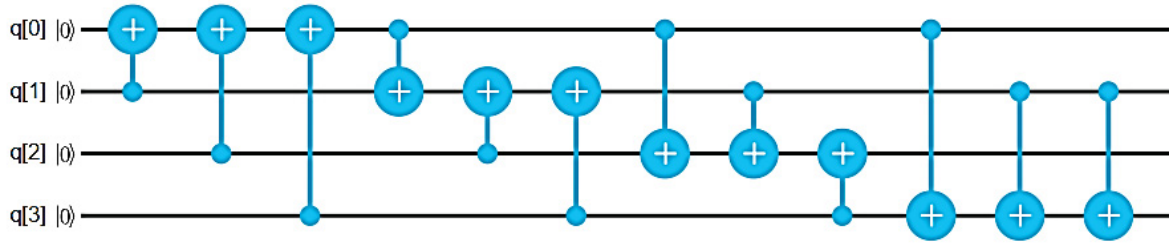
Para a utilização da DQN, foi necessário o desenvolvimento de um ambiente de interação com a técnica. Esse ambiente recebe uma ação de um agente, altera o estado do ambiente e retorna o novo estado e uma recompensa. O conjunto de ações possíveis é definido de forma semelhante ao mapeamento feito para as técnicas ANN

e CNN, diferindo apenas no caso de portas quânticas de 2 q-bits ou mais. O estado inicial é definido como um *grid* de um circuito, com uma quantidade de q-bits e operações quânticas definidas pelo usuário, com todas as posições preenchidas com operações identidade. O agente, então, vai preenchendo posição a posição do *grid*, preenchendo primeiro as colunas, que representam as operações, completando todos os q-bits de uma mesma operação quântica antes de seguir para a próxima. Esse procedimento é necessário devido às portas quânticas de 2 q-bits ou mais, como a CNOT, que precisa preencher dois espaços no *grid* em uma mesma operação, com uma única ação. Para portas quânticas de 2 q-bits ou mais, o número de ações possíveis, A, é igual ao número de arranjos simples entre a quantidade de q-bits, q, do circuito e quantos q-bits são afetados pela porta quântica, p, de modo que

$$A = \frac{q!}{(q-p)!}. \tag{4.1}$$

Como exemplo, uma porta CNOT pode ser disposta em doze maneiras diferentes em um circuito de 4 q-bits, como apresenta a FIGURA 24.

FIGURA 24 – ARRANJOS DE UMA PORTA CNOT (4-Q-BITS)



FONTE: O autor (2019).

Para as ações de inserção de portas quânticas de 1 q-bit, a operação é direta, ou seja, uma porta quântica é escolhida e inserida no *grid*, sequencialmente. Sendo assim, o número de ações possíveis, para portas de 1 q-bit, é igual à quantidade de portas quânticas disponíveis mais a operação nula. No caso do IBM-Q 5 Tenerife, oito portas quânticas de 1 q-bit estão disponíveis, totalizando nove operações possíveis. Portanto, o número máximo de ações, A_T, que o agente pode realizar em um IBM-Q 5 Tenerife, que possui 5 q-bits, oito portas quânticas de 1 q-bit e uma porta quântica de 2 q-bits (IBM, 2019), é

$$A_T = 8 + 1 + \frac{5!}{(5-2)!} = 29\,. \tag{4.2}$$

Com o conjunto de ações e estado inicial definidos, o processo de decisão de Markov se inicia. Partindo do estado inicial, o agente arbitra uma ação, que é a escolha de uma porta quântica para a primeira posição do *grid,* altera o estado inicial para o novo estado, ou seja, o estado inicial contendo a porta escolhida pela ação na posição relativa ao primeiro q-bit e primeira ação. Esse novo circuito é inserido como entrada do simulador de dispositivos quânticos que retorna a matriz resposta desse circuito. Essa resposta é, então, comparada com a matriz que representa o algoritmo que se deseja descobrir o circuito gerador, utilizando o somatório dos valores absolutos das diferenças, ε, entre os termos, μ, da matriz desejada e os termos, ρ, da matriz resposta, de forma que

$$\varepsilon = \sum_{i=1}^{N}\sum_{j=1}^{N}\left|\mu_{ij} - \rho_{ij}\right|, \tag{4.3}$$

onde N representa a quantidade de linhas e colunas das matrizes e i, j representam os índices das linhas e das colunas das matrizes, respectivamente. Para limitar os valores em um intervalo finito e converter essa medida de erro em uma medida de acurácia, foi adotada uma métrica de pontuação, Ω, sendo

$$\Omega = \frac{1}{1+\varepsilon}\,. \tag{4.4}$$

A imagem dessa métrica possui o intervalo $]0, 1]$, sendo que o valor unitário representa a igualdade entre as matrizes.

A métrica S é comparada com uma tolerância, que define para quais valores de Ω o agente deverá receber uma recompensa. Quanto mais próximo da unidade essa tolerância estiver, mais próximo da matriz desejada a matriz resposta do simulador deverá estar para que uma recompensa seja dada.

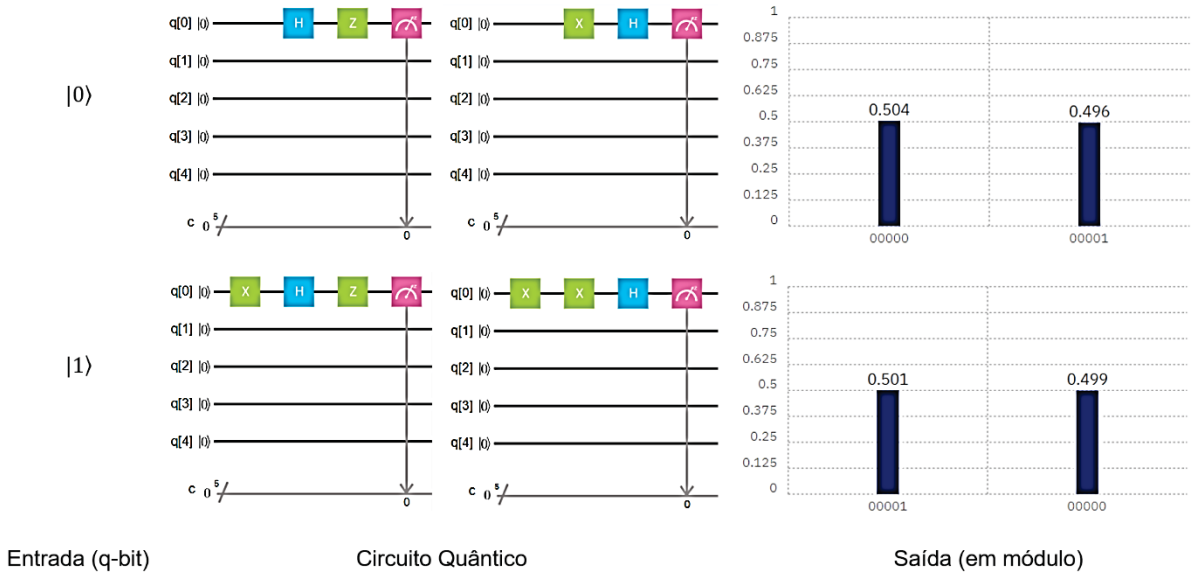
4.5 AJUSTES NA GERAÇÃO DE DADOS E TESTE DO *Q-LEARNING*

Diferente de tarefas ordinárias de classificação ou de regressão, o treinamento de uma rede neural artificial supervisionada, seja ela uma ANN ou CNN, para representar um compilador quântico, possui algumas características que precisam ser consideradas na criação do conjunto de dados para treinamento, já que o domínio da solução é a dos números complexos e um mesmo algoritmo quântico pode ser representado por mais de um circuito diferente, ou seja, não existe uma função genérica que, dado um algoritmo quântico qualquer, represente todos os circuitos que possuem esse algoritmo como solução. Por exemplo, o algoritmo quântico, representado pela matriz

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix},$$

pode ser gerado por dois circuitos distintos de 1 q-bits, apresentados na FIGURA 25.

FIGURA 25 – CIRCUITOS QUÂNTICOS COM A MESMA RESPOSTA



FONTE: Adaptado de IBM (2019).

A porta quântica inversora, X, foi adicionada à esquerda do circuito para simular o q-bit $|1\rangle$, já que o IBM-Q 5 Tenerife inicia todos os q-bits de entrada como

$|0\rangle$ (IBM, 2019). Assim sendo, todas as possibilidades de entradas para 1 q-bit foram analisadas, mostrando que os resultados são semelhantes. As diferenças entre os valores de saída se devem ao fato de que o resultado real é realizado com a distribuição real de uma quantidade finita de medições que, nesse caso, foi definida como 8.000 amostras. De forma analítica, a operação linear dos dois circuitos apresentados é descrita como:

$$ZH = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \cdot \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$$
$$HX = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}. \quad (4.5)$$

As operações das portas quânticas são invertidas devido à propriedade das operações lineares feitas sobre os q-bits de entrada, definidas no item 2.1.

4.5.1 Limitações dos dados para ANN e CNN

Como definido no item 3.1.1, uma ANN é formada por um conjunto de elementos, denominados neurônios artificiais que, por meio do ajuste de pesos, tendem a representar uma certa função desejada, $f(x_1, x_2, \ldots, x_n)$, onde o conjunto de variáveis $\{x_1, x_2, \ldots, x_n\}$ é utilizado como entrada desses neurônios. Assim sendo, uma rede neural artificial, em sua totalidade, representa uma função $f(x_1, x_2, \ldots, x_n)$, o que torna inviável a aplicação desse tipo de técnica como compilador quântico global, pois uma função de n variáveis é definida como uma regra que associa, a cada subconjunto de n números de um conjunto, um único valor denotado por $f(x_1, x_2, \ldots, x_n)$ (STEWART, 2013). É necessária, assim, a limitação de apenas um circuito para cada algoritmo quântico que se deseja representar, para que a compilação quântica possa ser representada por uma função, ou seja, admita somente um circuito quântico para cada algoritmo distinto. Essa adaptação também é necessária para a topologia da técnica CNN utilizada nesta dissertação, pois suas últimas camadas formam uma ANN.

Para tornar o conjunto de dados gerados pelo algoritmo descrito no item 4.2 uma função, uma lógica adicional foi inserida, onde cada novo circuito quântico aleatório válido gerado é comparado com todos os outros já incluídos no conjunto de

dados final. Caso haja algum circuito quântico nesse conjunto que possua a mesma resposta que o novo circuito gerado, esse último é substituído pelo já incluído e adicionado como uma nova amostra.

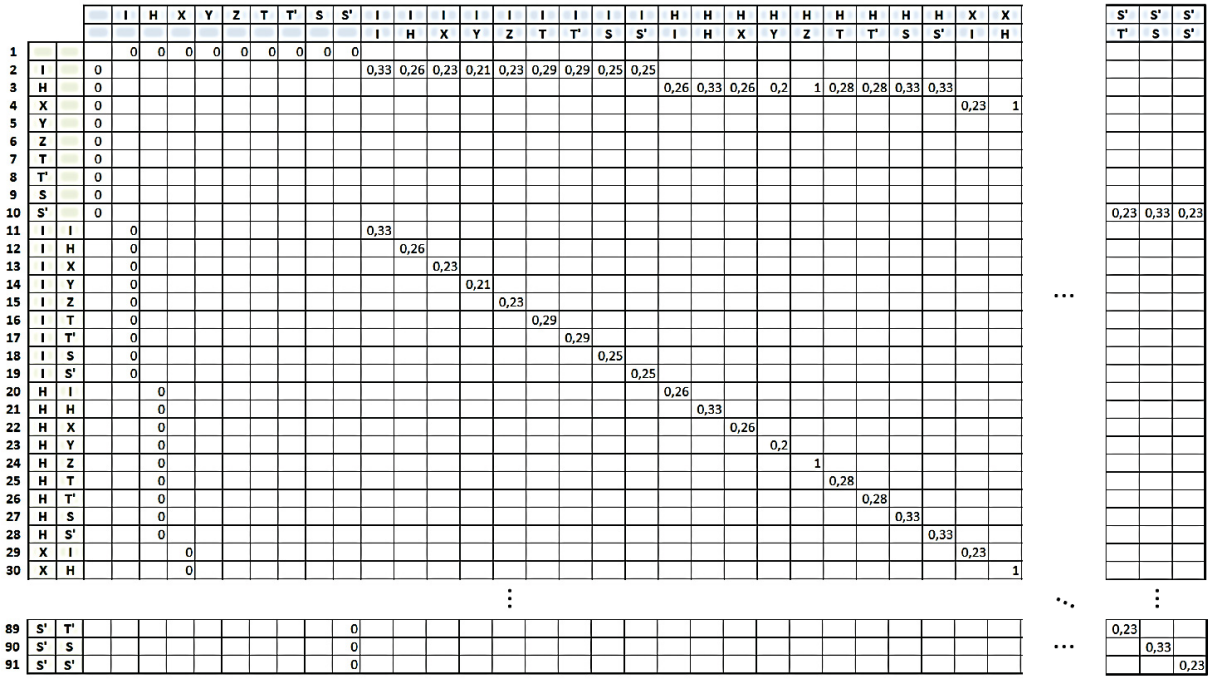
4.5.2 Teste de viabilidade da compilação quântica global

O problema da compilação quântica global não poder ser representada como função, por admitir mais de um circuito quântico para uma mesma resposta, não inviabiliza o uso de técnicas de Aprendizado de Máquina para essa aplicação, somente as técnicas supervisionadas, ou seja, que necessitem de um conjunto de dados que relacionem amostras de entrada com amostras de saída. Para testar se seria possível utilizar a DQN como técnica para a compilação quântica global, o *Q-Learning* foi aplicada à um circuito quântico de 1 q-bit para verificar se a técnica seria capaz de encontrar mais de um circuito quântico para um mesmo algoritmo, já que a DQN utilizada é, basicamente, um *Q-Learning* onde a tabela "Q" é definida por uma ANN multicamada. Utilizando novamente o algoritmo quântico, representado pela matriz

$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix},$$

que possui como circuitos as operações HZ e XH, como demonstrado no item 4.5, uma tabela de recompensas foi elaborada, manualmente, contendo todos os estados e ações possíveis pelo agente. As recompensas são definidas pelo valor de S, definido pela equação (4.4). A ação do agente se limita a colocar ou tirar uma porta quântica qualquer, uma por vez, limitado à duas portas totais no circuito. Com essas regras definidas, a tabela de recompensas foi preenchida, e parte dela é apresentada na FIGURA 26. As duas primeiras colunas representam o estado atual em que o agente se encontra, enquanto as duas primeiras linhas representam o estado futuro do agente de acordo com sua ação. A interseção entre o estado atual e o estado futuro representa a recompensa concedida para o agente quando ele realiza essa ação.

FIGURA 26 – TABELA DE RECOMPENSAS PARA O *Q-LEARNING*



FONTE: O autor (2019).

Por meio da aplicação dos conceitos descritos no item 3.1.3.3, foi possível concluir que a técnica *Q-Learning* é capaz de encontrar soluções diferentes cada vez que é treinada. Para esse teste específico, utilizando dez iterações e um fator de desconto de recompensa de 0,4, a técnica encontrou os dois circuitos possíveis, como apresentado na FIGURA 27.

FIGURA 27 – RESULTADOS DO TESTE PARA O *Q-LEARNING*

```
>> QLearning(R, 0.4, 10, 1)
Episode 1/10
Episode 2/10
Episode 3/10
Episode 4/10
Episode 5/10
Episode 6/10
Episode 7/10
Episode 8/10
Episode 9/10
Episode 10/10

ans =

     1     3    24
```

```
>> QLearning(R, 0.4, 10, 1)
Episode 1/10
Episode 2/10
Episode 3/10
Episode 4/10
Episode 5/10
Episode 6/10
Episode 7/10
Episode 8/10
Episode 9/10
Episode 10/10

ans =

     1     4    30
```

FONTE: O autor (2019).

A resposta apresentada é a sequência de estados, resultantes das ações do agente, até se chegar no circuito final. Cada estado é representado por um número, correspondente a uma linha da tabela de recompensas ilustrada na FIGURA 26. Assim sendo, a sequência no resultado à esquerda (1 – 3 – 24) da FIGURA 27 representa a sequência de estados percorridos pelo agente até chegar ao estado de índice 24, que representa o circuito HZ. Já a sequência no resultado à direita (1 – 4 – 30) é finalizada no índice 30, ou seja, o circuito XH.

4.6 AJUSTES ESPECÍFICOS DOS PARÂMETROS DE CADA TÉCNICA

As técnicas utilizadas necessitam do ajuste de alguns parâmetros, chamados hiperparâmetros, antes de serem utilizadas. Com o objetivo de diminuir o número de testes e comparações, alguns parâmetros foram mantidos fixos para as três técnicas, como o número de épocas, tamanho de *batch* e taxa de aprendizado.

Os itens a seguir apresentam as escolhas feitas para cada uma das técnicas de Aprendizado de Máquina utilizados.

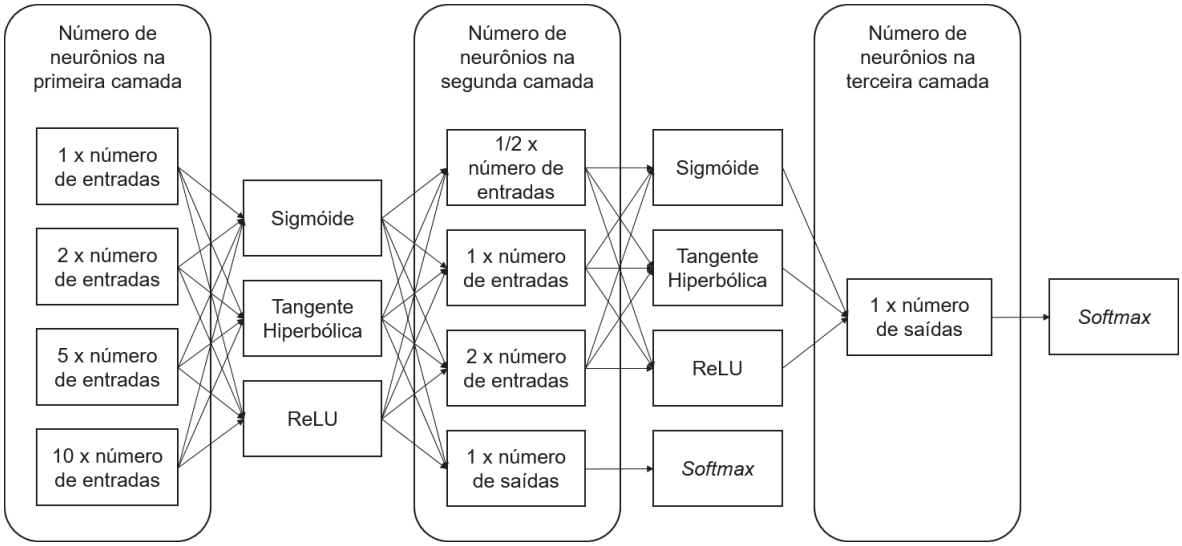
4.6.1 Rede neural artificial multicamada

Uma rede neural artificial multicamada possui vários hiperparâmetros que podem ser ajustados, dependendo do tipo de técnica de otimização escolhida, topologia e métricas. Dentre esses hiperparâmetros de controle, alguns são comuns à ANN, CNN e a DQN:

- Número de camadas;
- Número de neurônios em cada camada;
- Função de ativação dos neurônios;
- Métrica de erro.

As combinações apresentadas na FIGURA 28 foram utilizadas para comparação de resultados, com o objetivo de encontrar a melhor topologia para a resolução do problema da compilação quântica utilizando redes neurais artificiais.

FIGURA 28 – TOPOLOGIAS UTILIZADAS PARA A ANN



FONTE: O autor (2019).

O número de neurônios em cada camada foi definido como um múltiplo do número de entradas ou de saídas, arbitrariamente, com o objetivo de simplificar a automatização dos testes e comparar o efeito de quantidades diferentes, mas não específicas, de neurônios em cada camada. A função de ativação na última camada, de todas as topologias comparadas, foi definida como sendo a função *softmax*, definida como

$$\vartheta(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}, \tag{4.6}$$

onde a saída ϑ do termo de índice i do vetor z é igual ao exponencial natural do termo z_i sobre o somatório do exponencial natural de todos os outros termos contidos em z. A função *softmax* é uma forma de função logística que normaliza um valor de entrada em um vetor de valores que seguem uma distribuição probabilística, totalizando a unidade quando somados. Os valores de saída da função *softmax* se encontram no intervalo $[0, 1]$, o que permite uma quantidade dinâmica de classes categóricas (KARCZEWSKI et al., 2019). No caso deste trabalho, a função *softmax* permite que vários tamanhos de circuitos quânticos diferentes sejam utilizados sem a necessidade de alterar os conjuntos de dados.

Como função custo, a entropia cruzada foi escolhida, e também foi mantida fixa para simplificação. A entropia cruzada é definida como

$$\Gamma = -\frac{1}{N}\sum_{x}\sum_{j}\left[Y_j \ln a_j^l + \left(1 - Y_j\right)\ln\left(1 - a_j^l\right)\right], \tag{4.7}$$

onde N é o número total de itens no conjunto de dados de treinamento, o somatório externo é feito sobre todas as entradas de treinamento, x, e o somatório interno é feito sobre todos os neurônios, j, enquanto Y corresponde à saída desejada e a corresponde a saída atual do neurônio j na camada L. A entropia cruzada foi escolhida por evitar o problema de desaceleração na taxa de aprendizagem causada por outras funções custo, como é o caso do erro médio quadrático (NIELSEN, 2015).

Como métrica de acurácia, foi escolhida a métrica de pontuação personalizada, ε, definida no item 4.4, pela equação (4.4). A otimização do treinamento foi realizada com o otimizador ADAM, um método estocástico que requer somente gradientes de primeira ordem, assim como a descida de encosta (KINGMA; BA, 2015).

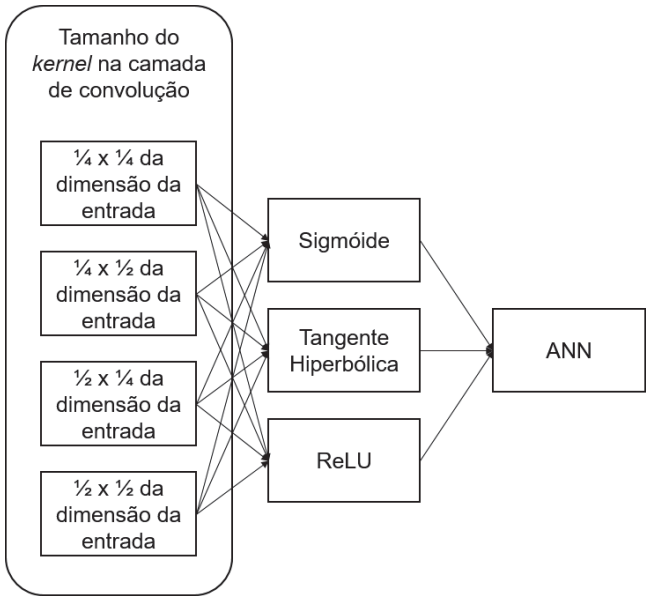
4.6.2 Rede neural convolucional

Além de possuir os hiperparâmetros da ANN, a rede neural convolucional possui um conjunto adicional de configurações referente às camadas convolucionais e de amostragem. No caso deste trabalho, não serão utilizadas camadas de amostragem, pois as dimensões das amostras de entrada não causam um custo computacional elevado, como normalmente acontece com imagens. Para limitar a quantidade de testes, a quantidade de camadas de convolução será pré-definida. Sendo assim, somente uma camada de convolução bidimensional e camadas totalmente conectadas serão utilizadas para a aplicação da técnica CNN. Os hiperparâmetros extras para as camadas de convolução bidimensionais são os seguintes:

- Número de características para cada camada de convolução;
- Tamanho vertical do *kernel* para cada camada de convolução;
- Tamanho horizontal do *kernel* para cada camada de convolução;
- Tamanho do passo vertical do *kernel* para cada camada de convolução;
- Tamanho do passo horizontal do *kernel* para cada camada de convolução.

Como são muitas combinações diferentes de hiperparâmetros, a topologia das camadas totalmente conectadas na rede neural convolucional foi definida como sendo igual à da ANN com o melhor resultado entre as topologias testadas no item 4.6.1. Assim sendo, foram testadas as configurações de hiperparâmetros nas camadas convolucionais bidimensionais apresentadas na FIGURA 29.

FIGURA 29 – TOPOLOGIAS UTILIZADAS PARA A CNN



FONTE: O autor (2019).

Para fins de simplificação, os passos do *kernel* foram mantidos fixos, sendo eles um passo unitário tanto no eixo vertical e um passo unitário no eixo horizontal. Todos os outros hiperparâmetros utilizados, referentes às camadas totalmente conectadas, foram mantidas as mesmas da ANN com o melhor desempenho. Nesse caso, a função de ativação da camada de saída foi mantida como sendo uma função *softmax*, o otimizador utilizado foi o ADAM e, como função custo, foi utilizada a entropia cruzada.

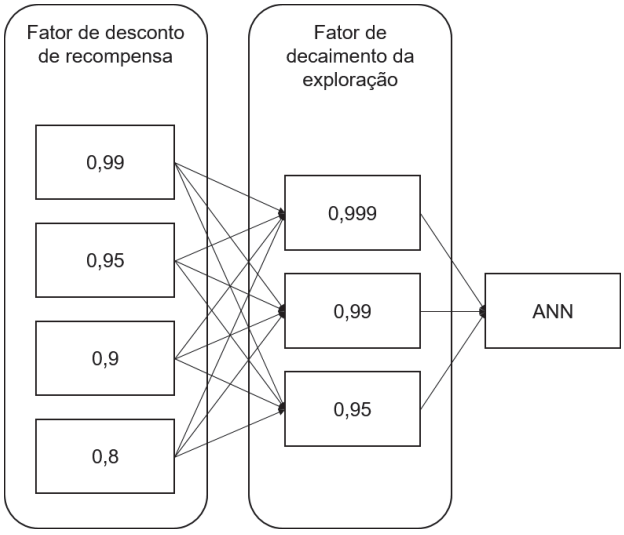
4.6.3 Deep Q-Network

Os mesmos hiperparâmetros ajustados para a ANN multicamadas também foram necessários no ajuste da DQN, pois foi a ANN a técnica utilizada para realizar o *Q-Learning*. Adicionalmente, a DQN precisa de alguns hiperparâmetros específicos da técnica. São eles:

- Fator de desconto de recompensa;
- Fator de exploração;
- Fator de exploração mínimo;
- Fator de decaimento da exploração.

O fator de exploração é uma probabilidade de que o agente realize uma ação aleatória. Conforme o aprendizado vai convergindo e o erro diminuindo, esse fator de exploração vai diminuindo, de acordo com o fator de decaimento da exploração, limitado ao valor definido pelo fator de exploração mínimo. Estes quatro fatores específicos da técnica DQN estão contidos no intervalo $[0, 1]$. Para fins de simplificação e redução do número de testes, o fator de exploração e o fator de exploração mínimo foram mantidos fixos. As configurações utilizadas para os testes do compilador quântico são ilustrados na FIGURA 30.

FIGURA 30 – AJUSTES DE PARÂMETROS UTILIZADOS PARA A DQN



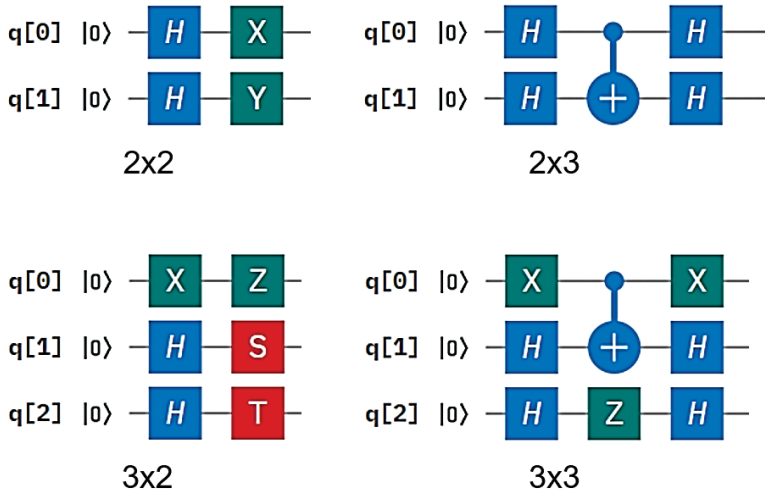
FONTE: O autor (2019).

Como a utilização da ANN multicamada dentro da DQN é diferente do que sua aplicação independente, a combinação analisada de função de ativação e função custo realizou uma tarefa de regressão, não sendo mais uma tarefa de classificação categórica, como no caso da ANN e da CNN. O motivo foi a simplificação do algoritmo e por apresentar resultados que permitiram a validação da técnica e comparação com as outras. A função de ativação, utilizada na camada de saída da ANN multicamada dentro da DQN, foi a função linear, enquanto a função custo foi definida como o erro médio quadrático.

5 RESULTADOS

Os conjuntos de dados gerados, contendo 40 000 amostras, foram divididos em 20 000 para os conjuntos de treinamento e 20 000 para os conjuntos de validação, tanto para os testes da ANN quanto para a CNN. O tamanho de *batch* e a quantidade de épocas também foram pré-definidos para limitar a quantidade de casos de teste, definidos como 30 e 50, respectivamente. Para fins de teste, foram adotadas quatro topologias diferentes de circuitos quânticos, sendo eles as combinações com 2 e 3 q-bits, para 2 e 3 operações quânticas. Um exemplo de circuito quântico para cada uma dessas topologias é apresentado na FIGURA 31.

FIGURA 31 – EXEMPLOS DE TOPOLOGIAS DE CIRCUITO UTILIZADAS PARA TESTE



FONTE: O autor (2019).

5.1 REDE NEURAL ARTIFICIAL MULTICAMADA

Com o objetivo de facilitar a visualização dos resultados, as tabelas foram agrupadas de acordo com o número de neurônios da primeira camada e a topologia do circuito respectivo ao teste. Assim sendo, os itens seguintes apresentam os resultados para as quatro topologias de circuitos quânticos apresentadas, e as tabelas contidas nesses itens estão divididas entre as combinações de neurônios de entrada apresentados na FIGURA 28, que são 1, 2, 5 e 10 vezes a quantidade de entradas propriamente ditas, ou seja, para um circuito com 2 q-bits, onde o resultado do circuito é sempre uma matriz complexa de 4x4, a quantidade de entradas será 32, pois, como apresentado no item 4.3, a parte real e imaginária da matriz é separada.

5.1.1 Resultados para circuitos 2x2

A TABELA 6 apresenta os resultados obtidos para as diferentes topologias propostas de ANN, apresentadas na FIGURA 28, para a compilação de circuitos quânticos de 2 q-bits com 2 operações quânticas, com 32 neurônios na primeira camada.

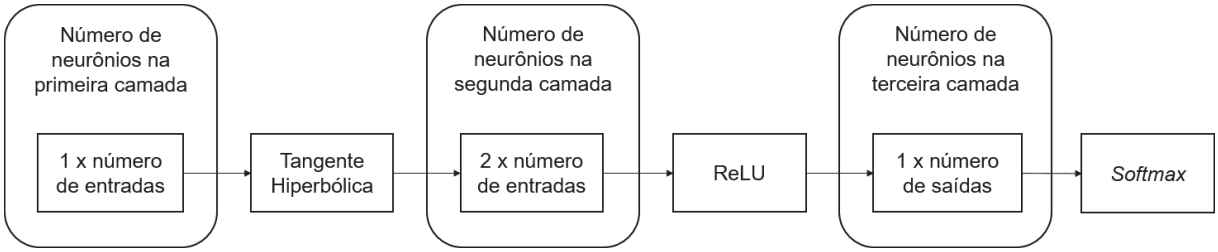
TABELA 6 – RESULTADOS PARA A ANN - CIRCUITOS 2X2 E 32 NEURÔNIOS DE ENTRADA

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
32, 16, 28	sigmoide, sigmoide, softmax	7,53	7,54	0,72750	95,54
32, 16, 28	sigmoide, tanh, softmax	7,23	7,25	0,81105	92,35
32, 16, 28	sigmoide, relu, softmax	7,67	7,70	0,71885	88,02
32, 32, 28	sigmoide, sigmoide, softmax	7,12	7,14	0,82280	87,15
32, 32, 28	sigmoide, tanh, softmax	6,84	6,86	0,86470	92,69
32, 32, 28	sigmoide, relu, softmax	7,06	7,07	0,83610	93,19
32, 64, 28	sigmoide, sigmoide, softmax	6,85	6,86	0,86645	86,56
32, 64, 28	sigmoide, tanh, softmax	6,64	6,65	0,91305	79,68
32, 64, 28	sigmoide, relu, softmax	6,62	6,64	0,91505	79,50
32, 28	sigmoide, softmax	7,44	7,46	0,76160	80,61
32, 16, 28	tanh, sigmoide, softmax	6,98	7,00	0,84620	81,11
32, 16, 28	tanh, tanh, softmax	6,76	6,79	0,89070	79,98
32, 16, 28	tanh, relu, softmax	6,70	6,73	0,89000	80,50
32, 32, 28	tanh, sigmoide, softmax	6,51	6,53	0,92910	81,46
32, 32, 28	tanh, tanh, softmax	6,27	6,31	0,96335	81,69
32, 32, 28	tanh, relu, softmax	6,23	6,27	0,96310	82,13
32, 64, 28	tanh, sigmoide, softmax	6,26	6,29	0,95460	83,58
32, 64, 28	tanh, tanh, softmax	6,08	6,12	0,98195	83,37
32, 64, 28	tanh, relu, softmax	6,04	6,08	**0,98725**	83,73
32, 28	tanh, softmax	6,78	6,80	0,89115	82,25
32, 16, 28	relu, sigmoide, softmax	6,86	6,88	0,85745	83,83
32, 16, 28	relu, tanh, softmax	6,59	6,62	0,91485	85,23
32, 16, 28	relu, relu, softmax	6,59	6,64	0,90625	89,51
32, 32, 28	relu, sigmoide, softmax	6,50	6,54	0,92310	85,32
32, 32, 28	relu, tanh, softmax	6,17	6,22	0,97215	85,43
32, 32, 28	relu, relu, softmax	6,25	6,30	0,95390	85,72
32, 64, 28	relu, sigmoide, softmax	6,22	6,26	0,96960	86,58
32, 64, 28	relu, tanh, softmax	6,08	6,11	0,98345	86,83
32, 64, 28	relu, relu, softmax	6,08	6,13	0,98590	87,66
32, 28	relu, softmax	6,63	6,67	0,89740	85,83

Fonte: O autor (2019).

A topologia que apresentou o melhor resultado, entre as combinações com 32 neurônios de entrada, foi a topologia apresentada na FIGURA 32. Para esta topologia, a acurácia foi de 0,98725, ou seja, 19.745 circuitos dos 20.000 contidos no conjunto de testes foram determinados corretamente.

FIGURA 32 – MELHOR RESULTADO ANN - CIRCUITOS 2X2 E 32 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 7 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 2 q-bits com 2 operações quânticas, com 64 neurônios na primeira camada.

TABELA 7 – RESULTADOS PARA A ANN - CIRCUITOS 2X2 E 64 NEURÔNIOS DE ENTRADA

(continua)

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
64, 16, 28	sigmoide, sigmoide, softmax	7,22	7,24	0,78230	87,78
64, 16, 28	sigmoide, tanh, softmax	7,11	7,13	0,83180	88,25
64, 16, 28	sigmoide, relu, softmax	7,98	7,97	0,67575	88,43
64, 32, 28	sigmoide, sigmoide, softmax	6,84	6,84	0,86450	88,73
64, 32, 28	sigmoide, tanh, softmax	6,54	6,56	0,93665	90,01
64, 32, 28	sigmoide, relu, softmax	6,82	6,86	0,87160	89,41
64, 64, 28	sigmoide, sigmoide, softmax	6,55	6,57	0,92420	91,22
64, 64, 28	sigmoide, tanh, softmax	6,34	6,38	0,95235	113,22
64, 64, 28	sigmoide, relu, softmax	6,58	6,60	0,92500	109,98
64, 16, 28	tanh, sigmoide, softmax	6,73	6,77	0,89305	108,69
64, 16, 28	tanh, tanh, softmax	6,68	6,71	0,90090	109,61
64, 16, 28	tanh, relu, softmax	6,49	6,53	0,92680	103,99
64, 32, 28	tanh, sigmoide, softmax	6,32	6,36	0,95610	94,09
64, 32, 28	tanh, tanh, softmax	6,17	6,21	0,97365	98,86
64, 32, 28	tanh, relu, softmax	6,12	6,18	0,97675	106,44
64, 64, 28	tanh, sigmoide, softmax	6,11	6,15	0,98135	96,60

TABELA 7 – RESULTADOS PARA A ANN - CIRCUITOS 2X2 E 64 NEURÔNIOS DE ENTRADA

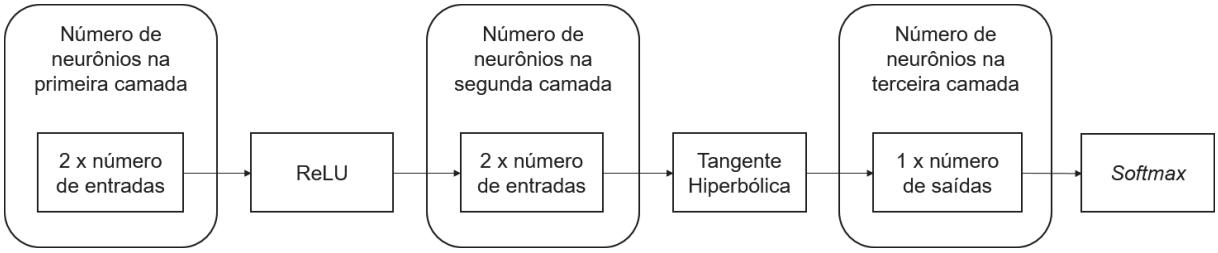
(conclusão)

Número de neurônios (1ª, 2ª, 3ª camada)	Funções de ativação (1ª, 2ª, 3ª camada)	MSE (treinamento)	MSE (validação)	Acurácia	Tempo de treinamento (s)
64, 64, 28	tanh, tanh, softmax	5,99	6,02	0,99755	96,68
64, 64, 28	tanh, relu, softmax	5,98	6,01	0,99880	97,80
64, 16, 28	relu, sigmoide, softmax	6,52	6,55	0,91800	97,31
64, 16, 28	relu, tanh, softmax	6,19	6,25	0,97505	97,15
64, 16, 28	relu, relu, softmax	6,22	6,28	0,97030	97,76
64, 32, 28	relu, sigmoide, softmax	6,18	6,21	0,97325	97,30
64, 32, 28	relu, tanh, softmax	6,00	6,04	0,99640	97,92
64, 32, 28	relu, relu, softmax	6,02	6,05	0,99280	98,52
64, 64, 28	relu, sigmoide, softmax	6,02	6,06	0,99235	99,80
64, 64, 28	relu, tanh, softmax	5,97	6,02	**0,99920**	100,99
64, 64, 28	relu, relu, softmax	5,97	6,01	0,99760	100,40

Fonte: O autor (2019).

Dentre as combinações com 64 neurônios de entrada, a topologia que apresentou o melhor resultado foi a apresentada na FIGURA 33, que obteve uma acurácia foi de 0,9992, representando 19.984 circuitos determinados corretamente dos 20.000 contidos no conjunto de testes.

FIGURA 33 – MELHOR RESULTADO ANN - CIRCUITOS 2X2 E 64 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 8 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 2 q-bits com 2 operações quânticas, com 160 neurônios na primeira camada.

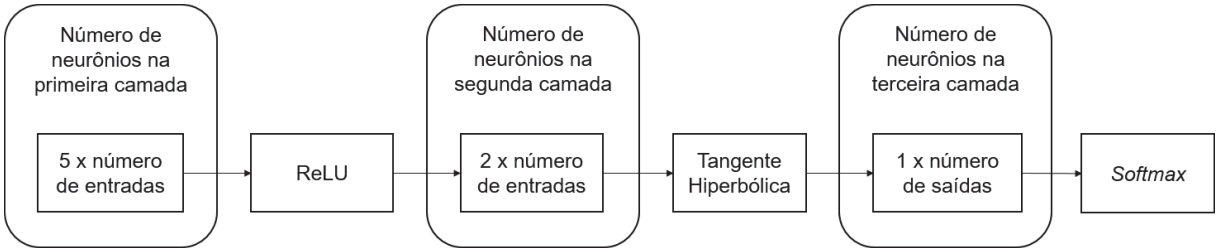
TABELA 8 – RESULTADOS PARA A ANN - CIRCUITOS 2X2 E 160 NEURÔNIOS DE ENTRADA

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
160, 16, 28	sigmoide, sigmoide, softmax	7,11	7,12	0,80975	100,88
160, 16, 28	sigmoide, tanh, softmax	7,14	7,16	0,82280	101,67
160, 16, 28	sigmoide, relu, softmax	8,92	8,92	0,47800	101,62
160, 32, 28	sigmoide, sigmoide, softmax	6,59	6,61	0,90865	102,83
160, 32, 28	sigmoide, tanh, softmax	6,41	6,44	0,94820	105,29
160, 32, 28	sigmoide, relu, softmax	7,95	7,94	0,66465	103,19
160, 64, 28	sigmoide, sigmoide, softmax	6,35	6,38	0,94665	105,19
160, 64, 28	sigmoide, tanh, softmax	6,18	6,22	0,97375	105,60
160, 64, 28	sigmoide, relu, softmax	6,57	6,60	0,90965	105,41
160, 16, 28	tanh, sigmoide, softmax	6,86	6,89	0,86280	104,90
160, 16, 28	tanh, tanh, softmax	6,78	6,82	0,87055	105,50
160, 16, 28	tanh, relu, softmax	6,39	6,43	0,94475	105,05
160, 32, 28	tanh, sigmoide, softmax	6,28	6,32	0,96230	106,33
160, 32, 28	tanh, tanh, softmax	6,14	6,18	0,97770	106,53
160, 32, 28	tanh, relu, softmax	6,06	6,10	0,98605	107,01
160, 64, 28	tanh, sigmoide, softmax	6,04	6,07	0,98665	109,70
160, 64, 28	tanh, tanh, softmax	5,97	6,01	0,99685	109,30
160, 64, 28	tanh, relu, softmax	5,98	6,04	0,99760	109,46
160, 16, 28	relu, sigmoide, softmax	6,27	6,30	0,96165	108,24
160, 16, 28	relu, tanh, softmax	6,01	6,05	0,99790	108,30
160, 16, 28	relu, relu, softmax	6,04	6,07	0,99535	108,82
160, 32, 28	relu, sigmoide, softmax	6,03	6,07	0,99390	109,68
160, 32, 28	relu, tanh, softmax	5,95	5,99	0,99635	109,94
160, 32, 28	relu, relu, softmax	5,97	6,00	0,99800	111,02
160, 64, 28	relu, sigmoide, softmax	5,95	5,98	0,99750	112,26
160, 64, 28	relu, tanh, softmax	5,95	5,98	**0,99985**	112,54
160, 64, 28	relu, relu, softmax	5,96	6,00	0,99660	113,76

Fonte: O autor (2019).

Considerando as combinações de topologias de ANN, com 160 neurônios de entrada, a que apresentou o melhor resultado foi a topologia apresentada na FIGURA 34, obtendo 19.997 circuitos dos 20.000 contidos no conjunto de testes corretamente, resultado representado por uma acurácia de 0,99985.

FIGURA 34 – MELHOR RESULTADO ANN - CIRCUITOS 2X2 E 160 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 9 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 2 q-bits com 2 operações quânticas, com 320 neurônios na primeira camada.

TABELA 9 – RESULTADOS PARA A ANN - CIRCUITOS 2X2 E 320 NEURÔNIOS DE ENTRADA

(continua)

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
320, 16, 28	sigmoide, sigmoide, softmax	7,22	7,23	0,80000	113,23
320, 16, 28	sigmoide, tanh, softmax	7,24	7,26	0,80255	113,87
320, 16, 28	sigmoide, relu, softmax	8,48	8,51	0,58810	114,40
320, 32, 28	sigmoide, sigmoide, softmax	6,55	6,58	0,90810	118,45
320, 32, 28	sigmoide, tanh, softmax	6,45	6,49	0,93580	116,79
320, 32, 28	sigmoide, relu, softmax	7,27	7,28	0,81435	116,53
320, 64, 28	sigmoide, sigmoide, softmax	6,23	6,26	0,96720	143,34
320, 64, 28	sigmoide, tanh, softmax	6,18	6,22	0,97755	144,16
320, 64, 28	sigmoide, relu, softmax	7,41	7,41	0,76070	153,39
320, 16, 28	tanh, sigmoide, softmax	7,11	7,12	0,81650	117,79
320, 16, 28	tanh, tanh, softmax	6,93	6,98	0,84330	118,17
320, 16, 28	tanh, relu, softmax	6,49	6,51	0,91655	118,39
320, 32, 28	tanh, sigmoide, softmax	6,38	6,41	0,94765	120,09
320, 32, 28	tanh, tanh, softmax	6,20	6,27	0,96775	120,20
320, 32, 28	tanh, relu, softmax	6,11	6,14	0,98670	121,11
320, 64, 28	tanh, sigmoide, softmax	6,04	6,07	0,99200	150,06
320, 64, 28	tanh, tanh, softmax	5,98	6,01	0,99820	156,64
320, 64, 28	tanh, relu, softmax	6,00	6,04	0,99675	150,26
320, 16, 28	relu, sigmoide, softmax	6,20	6,24	0,97670	121,99
320, 16, 28	relu, tanh, softmax	6,02	6,06	0,99765	122,25

TABELA 9 – RESULTADOS PARA A ANN - CIRCUITOS 2X2 E 320 NEURÔNIOS DE ENTRADA

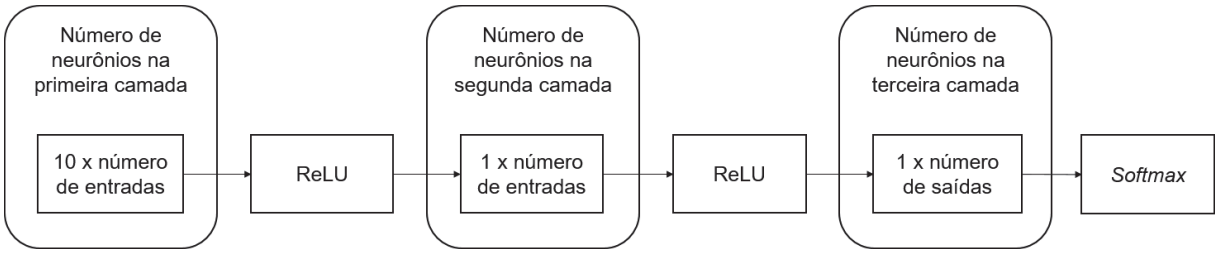
(conclusão)

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
320, 16, 28	relu, relu, softmax	5,99	6,02	0,99575	122,72
320, 32, 28	relu, sigmoide, softmax	5,98	6,01	0,99700	124,21
320, 32, 28	relu, tanh, softmax	5,95	5,99	0,99620	124,78
320, 32, 28	relu, relu, softmax	5,97	6,02	**0,99970**	**124,92**
320, 64, 28	relu, sigmoide, softmax	5,94	5,97	0,99825	159,50
320, 64, 28	relu, tanh, softmax	5,94	5,97	0,99810	161,84
320, 64, 28	relu, relu, softmax	5,96	6,01	**0,99970**	163,14

Fonte: O autor (2019).

O melhor resultado, representado pela maior acurácia e menor custo computacional, entre as combinações de topologias com 320 neurônios de entrada, para a técnica ANN, foi a topologia ilustrada na FIGURA 35, pela qual foi obtida uma acurácia de 0,9997, representando 19.994 circuitos dos 20.000 contidos no conjunto de testes determinados corretamente.

FIGURA 35 – MELHOR RESULTADO ANN - CIRCUITOS 2X2 E 320 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

Comparando-se os resultados obtidos nos quatro testes, o melhor resultado para a compilação de algoritmos quânticos, de 2 q-bits com 2 operações quânticas, foi obtido pela topologia com 160 neurônios na primeira camada, 64 neurônios na segunda camada e 28 neurônios na camada de saída, com funções de ativação RELU, tangente hiperbólica e *softmax*, respectivamente, com uma acurácia de 0,99985, ou seja, 19.997 circuitos dos 20.000 contidos no conjunto de testes foram determinados corretamente.

5.1.2 Resultados para circuitos 2x3

A TABELA 10 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 2 q-bits com 3 operações quânticas, com 32 neurônios na primeira camada.

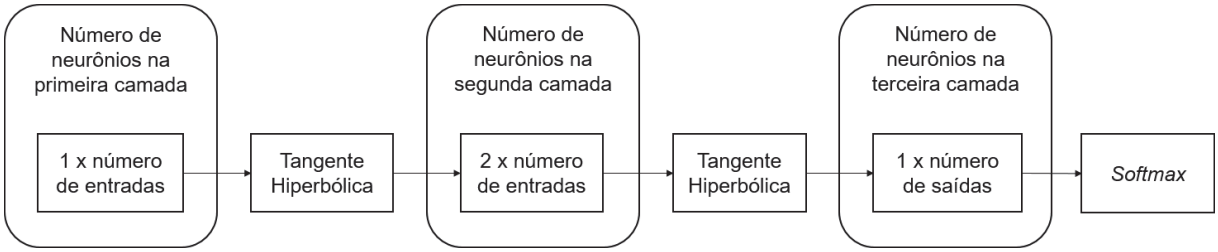
TABELA 10 – RESULTADOS PARA A ANN - CIRCUITOS 2X3 E 32 NEURÔNIOS DE ENTRADA

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
32, 16, 42	sigmoide, sigmoide, softmax	16,11	16,09	0,45620	91,07
32, 16, 42	sigmoide, tanh, softmax	16,18	16,17	0,40995	93,05
32, 16, 42	sigmoide, relu, softmax	16,32	16,31	0,43355	96,31
32, 32, 42	sigmoide, sigmoide, softmax	15,80	15,78	0,49165	96,26
32, 32, 42	sigmoide, tanh, softmax	15,62	15,61	0,55215	99,08
32, 32, 42	sigmoide, relu, softmax	15,65	15,64	0,52210	131,65
32, 64, 42	sigmoide, sigmoide, softmax	15,51	15,48	0,53460	133,69
32, 64, 42	sigmoide, tanh, softmax	15,44	15,42	0,56745	149,43
32, 64, 42	sigmoide, relu, softmax	15,31	15,33	0,58760	137,74
32, 42	sigmoide, softmax	16,23	16,22	0,45435	136,79
32, 16, 42	tanh, sigmoide, softmax	15,55	15,53	0,51960	141,18
32, 16, 42	tanh, tanh, softmax	15,63	15,65	0,54225	143,68
32, 16, 42	tanh, relu, softmax	15,51	15,52	0,52610	141,14
32, 32, 42	tanh, sigmoide, softmax	15,10	15,09	0,61470	144,34
32, 32, 42	tanh, tanh, softmax	14,91	14,97	0,62030	146,73
32, 32, 42	tanh, relu, softmax	14,97	15,01	0,60800	144,87
32, 64, 42	tanh, sigmoide, softmax	14,63	14,66	0,64470	145,88
32, 64, 42	tanh, tanh, softmax	14,32	14,40	**0,71850**	123,40
32, 64, 42	tanh, relu, softmax	14,38	14,47	0,69880	100,88
32, 42	tanh, softmax	15,74	15,75	0,51800	97,82
32, 16, 42	relu, sigmoide, softmax	15,53	15,50	0,53775	100,03
32, 16, 42	relu, tanh, softmax	15,37	15,41	0,52115	100,75
32, 16, 42	relu, relu, softmax	15,38	15,42	0,53745	109,43
32, 32, 42	relu, sigmoide, softmax	14,99	15,00	0,60265	122,04
32, 32, 42	relu, tanh, softmax	14,88	14,93	0,62330	120,40
32, 32, 42	relu, relu, softmax	14,81	14,83	0,63250	116,70
32, 64, 42	relu, sigmoide, softmax	14,60	14,62	0,66290	113,10
32, 64, 42	relu, tanh, softmax	14,45	14,52	0,69050	104,47
32, 64, 42	relu, relu, softmax	14,31	14,42	0,71185	110,02
32, 42	relu, softmax	15,53	15,56	0,56140	108,11

Fonte: O autor (2019).

Considerando as combinações de topologias de ANN, com 32 neurônios de entrada, a que apresentou o melhor resultado foi a topologia apresentada na FIGURA 36, obtendo 14.370 circuitos dos 20.000 contidos no conjunto de testes corretamente, resultado representado por uma acurácia de 0,7185.

FIGURA 36 – MELHOR RESULTADO ANN - CIRCUITOS 2X3 E 32 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 11 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 2 q-bits com 3 operações quânticas, com 64 neurônios na primeira camada.

TABELA 11 – RESULTADOS PARA A ANN - CIRCUITOS 2X3 E 64 NEURÔNIOS DE ENTRADA

(continua)

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
64, 16, 42	sigmoide, sigmoide, softmax	15,92	15,90	0,44990	114,19
64, 16, 42	sigmoide, tanh, softmax	15,98	15,96	0,46310	116,92
64, 16, 42	sigmoide, relu, softmax	16,31	16,27	0,39640	114,53
64, 32, 42	sigmoide, sigmoide, softmax	15,50	15,48	0,53210	115,30
64, 32, 42	sigmoide, tanh, softmax	15,32	15,32	0,57585	117,07
64, 32, 42	sigmoide, relu, softmax	15,60	15,60	0,55590	117,58
64, 64, 42	sigmoide, sigmoide, softmax	15,15	15,14	0,58910	125,65
64, 64, 42	sigmoide, tanh, softmax	14,96	14,98	0,61570	121,77
64, 64, 42	sigmoide, relu, softmax	15,15	15,16	0,59140	121,43
64, 16, 42	tanh, sigmoide, softmax	15,45	15,45	0,53475	121,36
64, 16, 42	tanh, tanh, softmax	15,39	15,41	0,54085	121,42
64, 16, 42	tanh, relu, softmax	15,19	15,21	0,57375	124,59
64, 32, 42	tanh, sigmoide, softmax	14,82	14,84	0,62430	106,58
64, 32, 42	tanh, tanh, softmax	14,60	14,64	0,67340	122,19
64, 32, 42	tanh, relu, softmax	14,65	14,68	0,65085	119,51
64, 64, 42	tanh, sigmoide, softmax	14,31	14,36	0,71390	115,84

TABELA 11 – RESULTADOS PARA A ANN - CIRCUITOS 2X3 E 64 NEURÔNIOS DE ENTRADA

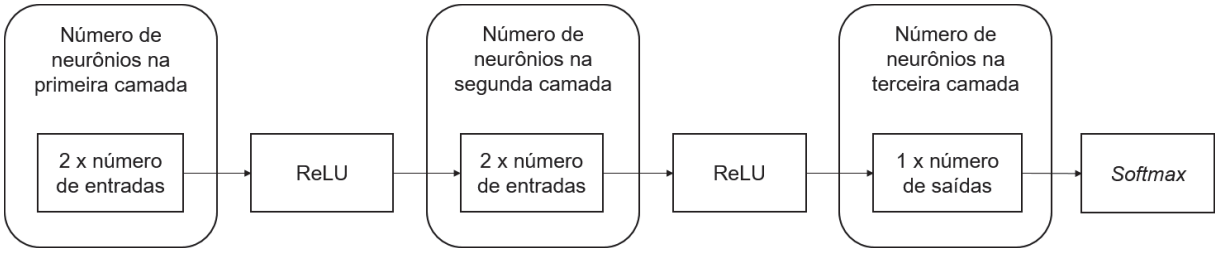
(conclusão)

Número de neurônios (1ª, 2ª, 3ª camada)	Funções de ativação (1ª, 2ª, 3ª camada)	MSE (treinamento)	MSE (validação)	Acurácia	Tempo de treinamento (s)
64, 64, 42	tanh, tanh, softmax	13,95	14,10	0,75500	112,92
64, 64, 42	tanh, relu, softmax	14,10	14,22	0,73315	113,65
64, 16, 42	relu, sigmoide, softmax	15,15	15,16	0,55545	111,02
64, 16, 42	relu, tanh, softmax	14,86	14,93	0,61575	112,37
64, 16, 42	relu, relu, softmax	14,81	14,85	0,60120	110,86
64, 32, 42	relu, sigmoide, softmax	14,54	14,58	0,66425	113,63
64, 32, 42	relu, tanh, softmax	14,30	14,41	0,70490	115,16
64, 32, 42	relu, relu, softmax	14,27	14,37	0,70540	113,20
64, 64, 42	relu, sigmoide, softmax	14,12	14,20	0,73370	115,87
64, 64, 42	relu, tanh, softmax	13,92	14,05	0,75250	116,48
64, 64, 42	relu, relu, softmax	13,83	14,03	**0,75770**	118,80

Fonte: O autor (2019).

Dentre as combinações com 64 neurônios de entrada, a topologia que apresentou o melhor resultado foi a apresentada na FIGURA 37, que obteve uma acurácia foi de 0,7577, representando 15.154 circuitos determinados corretamente dos 20.000 contidos no conjunto de testes.

FIGURA 37 – MELHOR RESULTADO ANN - CIRCUITOS 2X3 E 64 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 12 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 2 q-bits com 3 operações quânticas, com 160 neurônios na primeira camada.

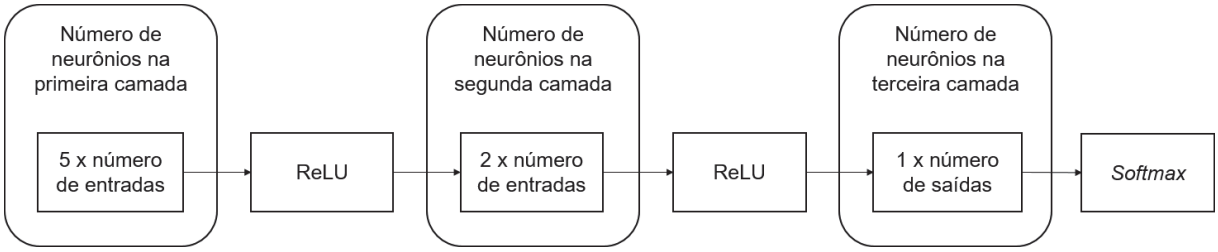
TABELA 12 – RESULTADOS PARA A ANN - CIRCUITOS 2X3 E 160 NEURÔNIOS DE ENTRADA

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
160, 16, 42	sigmoide, sigmoide, softmax	15,85	15,84	0,51155	120,82
160, 16, 42	sigmoide, tanh, softmax	16,12	16,09	0,44560	120,51
160, 16, 42	sigmoide, relu, softmax	16,95	16,94	0,33455	120,78
160, 32, 42	sigmoide, sigmoide, softmax	15,21	15,19	0,56970	122,33
160, 32, 42	sigmoide, tanh, softmax	15,19	15,21	0,58900	121,95
160, 32, 42	sigmoide, relu, softmax	16,25	16,22	0,41225	121,78
160, 64, 42	sigmoide, sigmoide, softmax	14,76	14,78	0,62745	118,18
160, 64, 42	sigmoide, tanh, softmax	14,59	14,62	0,67580	118,72
160, 64, 42	sigmoide, relu, softmax	15,81	15,79	0,48125	123,25
160, 16, 42	tanh, sigmoide, softmax	15,48	15,47	0,52870	123,90
160, 16, 42	tanh, tanh, softmax	15,52	15,53	0,51010	125,37
160, 16, 42	tanh, relu, softmax	15,05	15,06	0,57940	123,10
160, 32, 42	tanh, sigmoide, softmax	14,68	14,71	0,63435	124,92
160, 32, 42	tanh, tanh, softmax	14,56	14,65	0,67440	124,33
160, 32, 42	tanh, relu, softmax	14,46	14,50	0,68870	125,61
160, 64, 42	tanh, sigmoide, softmax	14,00	14,09	0,74535	126,20
160, 64, 42	tanh, tanh, softmax	13,80	13,95	0,77735	129,52
160, 64, 42	tanh, relu, softmax	13,94	14,05	0,74785	127,20
160, 16, 42	relu, sigmoide, softmax	14,64	14,70	0,63980	127,18
160, 16, 42	relu, tanh, softmax	14,12	14,26	0,70560	127,90
160, 16, 42	relu, relu, softmax	14,19	14,29	0,70215	128,62
160, 32, 42	relu, sigmoide, softmax	13,97	14,05	0,73695	118,38
160, 32, 42	relu, tanh, softmax	13,55	13,73	0,79080	123,28
160, 32, 42	relu, relu, softmax	13,64	13,79	0,78285	134,66
160, 64, 42	relu, sigmoide, softmax	13,51	13,65	0,78345	122,24
160, 64, 42	relu, tanh, softmax	13,24	13,50	0,81690	122,52
160, 64, 42	relu, relu, softmax	13,22	13,47	**0,82595**	122,57

Fonte: O autor (2019).

A topologia que apresentou o melhor resultado, entre as combinações com 160 neurônios de entrada, foi a topologia apresentada na FIGURA 38. Para esta topologia, a acurácia foi de 0,82595, ou seja, 16.519 circuitos dos 20.000 contidos no conjunto de testes foram determinados corretamente.

FIGURA 38 – MELHOR RESULTADO ANN - CIRCUITOS 2X3 E 160 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 13 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 2 q-bits com 3 operações quânticas, com 320 neurônios na primeira camada.

TABELA 13 – RESULTADOS PARA A ANN - CIRCUITOS 2X3 E 320 NEURÔNIOS DE ENTRADA

(continua)

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
320, 16, 42	sigmoide, sigmoide, softmax	15,90	15,90	0,47335	122,05
320, 16, 42	sigmoide, tanh, softmax	16,13	16,13	0,46080	121,97
320, 16, 42	sigmoide, relu, softmax	17,20	17,17	0,16450	123,28
320, 32, 42	sigmoide, sigmoide, softmax	15,13	15,14	0,57715	125,75
320, 32, 42	sigmoide, tanh, softmax	15,30	15,30	0,55245	125,77
320, 32, 42	sigmoide, relu, softmax	16,82	16,81	0,31040	125,76
320, 64, 42	sigmoide, sigmoide, softmax	14,56	14,59	0,66440	153,16
320, 64, 42	sigmoide, tanh, softmax	14,63	14,67	0,67730	153,94
320, 64, 42	sigmoide, relu, softmax	15,57	15,57	0,55645	163,90
320, 16, 42	tanh, sigmoide, softmax	15,62	15,65	0,49445	142,61
320, 16, 42	tanh, tanh, softmax	15,64	15,65	0,51245	142,25
320, 16, 42	tanh, relu, softmax	15,19	15,20	0,57230	143,46
320, 32, 42	tanh, sigmoide, softmax	14,74	14,79	0,65330	144,78
320, 32, 42	tanh, tanh, softmax	14,75	14,82	0,64275	145,77
320, 32, 42	tanh, relu, softmax	14,54	14,56	0,67850	171,70
320, 64, 42	tanh, sigmoide, softmax	13,97	14,07	0,74875	216,14
320, 64, 42	tanh, tanh, softmax	13,80	13,96	0,77020	217,68
320, 64, 42	tanh, relu, softmax	14,00	14,15	0,74235	203,83
320, 16, 42	relu, sigmoide, softmax	14,29	14,36	0,63945	182,04
320, 16, 42	relu, tanh, softmax	13,75	13,93	0,75970	208,33

TABELA 13 – RESULTADOS PARA A ANN - CIRCUITOS 2X3 E 320 NEURÔNIOS DE ENTRADA

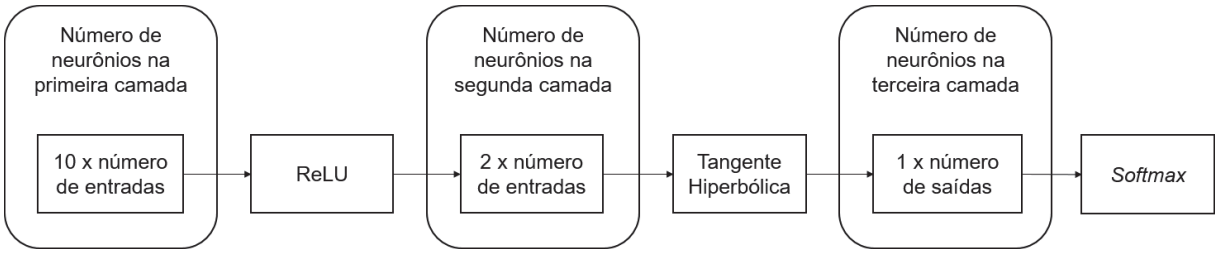
(conclusão)

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
320, 16, 42	relu, relu, softmax	13,79	13,95	0,73680	153,32
320, 32, 42	relu, sigmoide, softmax	13,62	13,76	0,76510	153,56
320, 32, 42	relu, tanh, softmax	13,16	13,44	0,82130	155,02
320, 32, 42	relu, relu, softmax	13,22	13,46	0,82695	156,68
320, 64, 42	relu, sigmoide, softmax	13,17	13,34	0,82575	188,23
320, 64, 42	relu, tanh, softmax	12,84	13,15	**0,85630**	188,77
320, 64, 42	relu, relu, softmax	12,88	13,18	0,85040	189,50

Fonte: O autor (2019).

O melhor resultado, representado pela maior acurácia, entre as combinações de topologias com 320 neurônios de entrada, para a técnica ANN, foi a topologia ilustrada na FIGURA 39, pela qual foi obtida uma acurácia de 0,8563, representando 17.126 circuitos dos 20.000 contidos no conjunto de testes determinados corretamente.

FIGURA 39 – MELHOR RESULTADO ANN - CIRCUITOS 2X3 E 320 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

Comparando-se os resultados obtidos nos quatro testes, o melhor resultado para a compilação de algoritmos quânticos, de 2 q-bits com 3 operações quânticas, foi obtido pela topologia com 320 neurônios na primeira camada, 64 neurônios na segunda camada e 42 neurônios na camada de saída, com funções de ativação RELU, tangente hiperbólica e *softmax*, respectivamente, com uma acurácia de 0,8563, ou seja, 17.126 circuitos dos 20.000 contidos no conjunto de testes foram determinados corretamente.

5.1.3 Resultados para circuitos 3x2

A TABELA 14 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 3 q-bits com 2 operações quânticas, com 128 neurônios na primeira camada.

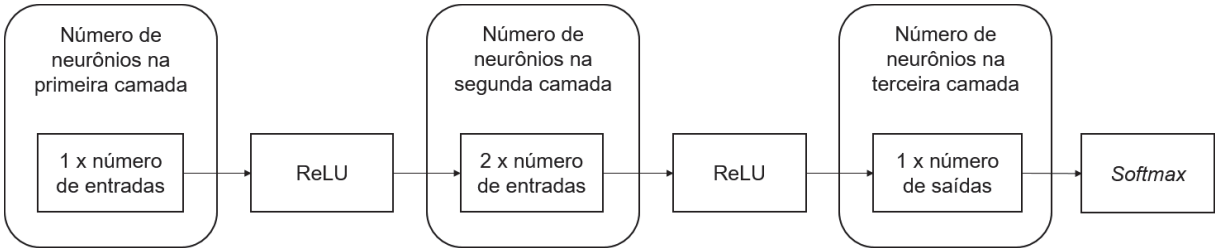
TABELA 14 – RESULTADOS PARA A ANN - CIRCUITOS 3X2 E 128 NEURÔNIOS DE ENTRADA

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
128, 64, 42	sigmoide, sigmoide, softmax	14,26	14,33	0,63620	89,26
128, 64, 42	sigmoide, tanh, softmax	14,05	14,18	0,66560	89,76
128, 64, 42	sigmoide, relu, softmax	15,04	15,09	0,56315	89,62
128, 128, 42	sigmoide, sigmoide, softmax	13,99	14,07	0,67510	92,76
128, 128, 42	sigmoide, tanh, softmax	13,85	14,01	0,68320	103,17
128, 128, 42	sigmoide, relu, softmax	14,10	14,23	0,65990	95,83
128, 256, 42	sigmoide, sigmoide, softmax	13,79	13,89	0,70360	99,87
128, 256, 42	sigmoide, tanh, softmax	13,89	14,02	0,68330	109,29
128, 256, 42	sigmoide, relu, softmax	13,66	13,85	0,70760	112,57
128, 42	sigmoide, softmax	15,10	15,18	0,55620	100,98
128, 64, 42	tanh, sigmoide, softmax	13,31	13,56	0,74040	99,42
128, 64, 42	tanh, tanh, softmax	13,06	13,45	0,76325	100,33
128, 64, 42	tanh, relu, softmax	13,14	13,51	0,76980	97,04
128, 128, 42	tanh, sigmoide, softmax	12,85	13,14	0,79255	106,16
128, 128, 42	tanh, tanh, softmax	12,55	13,04	0,82445	107,76
128, 128, 42	tanh, relu, softmax	12,57	13,07	0,82545	106,18
128, 256, 42	tanh, sigmoide, softmax	12,56	12,89	0,82450	111,74
128, 256, 42	tanh, tanh, softmax	12,22	12,71	0,87135	111,69
128, 256, 42	tanh, relu, softmax	12,17	12,75	0,87510	116,14
128, 42	tanh, softmax	13,79	14,01	0,70070	106,56
128, 64, 42	relu, sigmoide, softmax	12,90	13,26	0,77915	113,20
128, 64, 42	relu, tanh, softmax	12,63	13,17	0,81940	110,47
128, 64, 42	relu, relu, softmax	12,63	13,15	0,81275	110,32
128, 128, 42	relu, sigmoide, softmax	12,68	13,09	0,80905	113,96
128, 128, 42	relu, tanh, softmax	12,54	13,13	0,83380	113,61
128, 128, 42	relu, relu, softmax	12,31	12,85	0,85405	113,34
128, 256, 42	relu, sigmoide, softmax	12,56	12,97	0,82640	114,60
128, 256, 42	relu, tanh, softmax	12,56	13,15	0,83140	116,41
128, 256, 42	relu, relu, softmax	12,04	12,65	**0,88690**	119,48
128, 42	relu, softmax	13,04	13,44	0,77510	111,21

Fonte: O autor (2019).

Considerando as combinações de topologias de ANN, com 128 neurônios de entrada, a que apresentou o melhor resultado foi a topologia apresentada na FIGURA 40, obtendo 17.738 circuitos dos 20.000 contidos no conjunto de testes corretamente, resultado representado por uma acurácia de 0,8869.

FIGURA 40 – MELHOR RESULTADO ANN - CIRCUITOS 3X2 E 128 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 15 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 3 q-bits com 2 operações quânticas, com 256 neurônios na primeira camada.

TABELA 15 – RESULTADOS PARA A ANN - CIRCUITOS 3X2 E 256 NEURÔNIOS DE ENTRADA

(continua)

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
256, 64, 42	sigmoide, sigmoide, softmax	14,03	14,14	0,66080	120,97
256, 64, 42	sigmoide, tanh, softmax	13,83	14,00	0,69145	119,88
256, 64, 42	sigmoide, relu, softmax	14,86	14,94	0,54490	121,31
256, 128, 42	sigmoide, sigmoide, softmax	13,63	13,76	0,69980	122,70
256, 128, 42	sigmoide, tanh, softmax	13,46	13,67	0,72965	124,40
256, 128, 42	sigmoide, relu, softmax	14,16	14,29	0,64910	123,64
256, 256, 42	sigmoide, sigmoide, softmax	13,37	13,53	0,74105	134,50
256, 256, 42	sigmoide, tanh, softmax	13,31	13,46	0,74790	133,33
256, 256, 42	sigmoide, relu, softmax	13,68	13,90	0,70490	135,63
256, 64, 42	tanh, sigmoide, softmax	13,12	13,43	0,75850	126,29
256, 64, 42	tanh, tanh, softmax	13,03	13,48	0,77820	125,08
256, 64, 42	tanh, relu, softmax	13,03	13,41	0,77230	117,65
256, 128, 42	tanh, sigmoide, softmax	12,60	12,96	0,82365	121,00
256, 128, 42	tanh, tanh, softmax	12,36	12,91	0,84315	126,27
256, 128, 42	tanh, relu, softmax	12,51	13,07	0,83760	129,98
256, 256, 42	tanh, sigmoide, softmax	12,22	12,63	0,86930	134,70

TABELA 15 – RESULTADOS PARA A ANN - CIRCUITOS 3X2 E 256 NEURÔNIOS DE ENTRADA

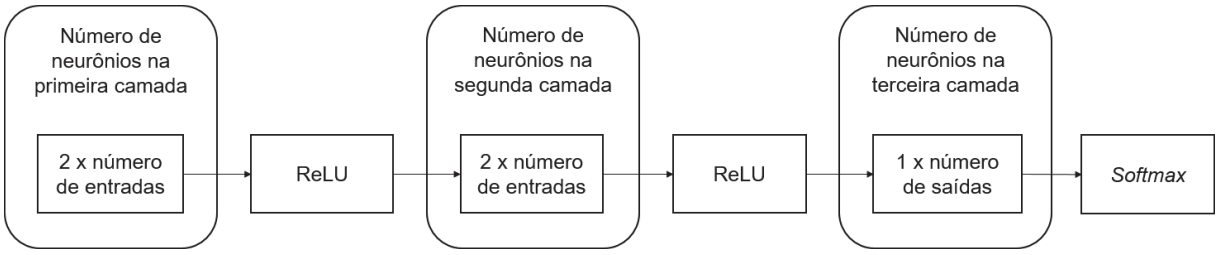
(conclusão)

Número de neurônios (1ª, 2ª, 3ª camada)	Funções de ativação (1ª, 2ª, 3ª camada)	MSE (treinamento)	MSE (validação)	Acurácia	Tempo de treinamento (s)
256, 256, 42	tanh, tanh, softmax	11,99	12,61	0,89355	134,31
256, 256, 42	tanh, relu, softmax	12,13	12,77	0,88280	132,35
256, 64, 42	relu, sigmoide, softmax	12,34	12,80	0,84790	135,81
256, 64, 42	relu, tanh, softmax	12,08	12,77	0,87975	132,97
256, 64, 42	relu, relu, softmax	12,13	12,76	0,87990	136,73
256, 128, 42	relu, sigmoide, softmax	12,13	12,66	0,87605	136,82
256, 128, 42	relu, tanh, softmax	12,02	12,71	0,89040	138,49
256, 128, 42	relu, relu, softmax	11,96	12,63	0,89580	143,51
256, 256, 42	relu, sigmoide, softmax	12,05	12,62	0,88990	155,93
256, 256, 42	relu, tanh, softmax	12,07	12,77	0,89185	156,95
256, 256, 42	relu, relu, softmax	11,83	12,51	**0,91625**	153,69

Fonte: O autor (2019).

Dentre as combinações com 256 neurônios de entrada, a topologia que apresentou o melhor resultado foi a apresentada na FIGURA 41, que obteve uma acurácia foi de 0,91625, representando 18.325 circuitos determinados corretamente dos 20.000 contidos no conjunto de testes.

FIGURA 41 – MELHOR RESULTADO ANN - CIRCUITOS 3X2 E 256 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 16 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 3 q-bits com 2 operações quânticas, com 640 neurônios na primeira camada.

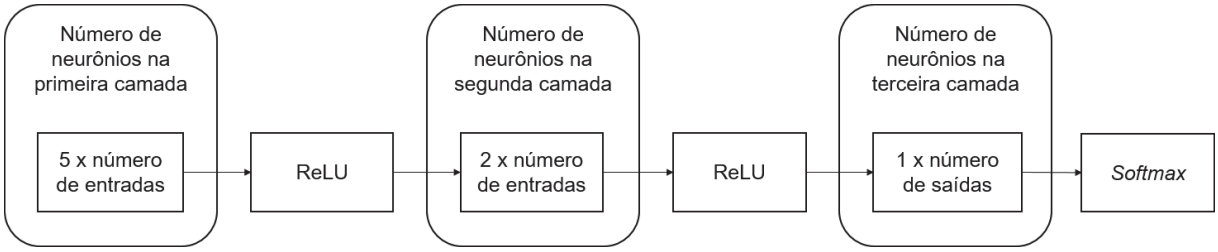
TABELA 16 – RESULTADOS PARA A ANN - CIRCUITOS 3X2 E 640 NEURÔNIOS DE ENTRADA

Número de neurônios (1ª, 2ª, 3ª camada)	Funções de ativação (1ª, 2ª, 3ª camada)	MSE (treinamento)	MSE (validação)	Acurácia	Tempo de treinamento (s)
640, 64, 42	sigmoide, sigmoide, softmax	13,75	13,92	0,68205	163,19
640, 64, 42	sigmoide, tanh, softmax	14,17	14,39	0,65195	162,16
640, 64, 42	sigmoide, relu, softmax	15,25	15,27	0,50890	163,87
640, 128, 42	sigmoide, sigmoide, softmax	13,30	13,50	0,74135	167,71
640, 128, 42	sigmoide, tanh, softmax	13,42	13,65	0,73200	158,45
640, 128, 42	sigmoide, relu, softmax	14,36	14,53	0,63975	160,98
640, 256, 42	sigmoide, sigmoide, softmax	12,92	13,15	0,79130	177,90
640, 256, 42	sigmoide, tanh, softmax	13,11	13,34	0,77145	172,25
640, 256, 42	sigmoide, relu, softmax	13,95	14,12	0,67415	176,49
640, 64, 42	tanh, sigmoide, softmax	13,18	13,56	0,75520	158,06
640, 64, 42	tanh, tanh, softmax	13,15	13,61	0,75320	155,92
640, 64, 42	tanh, relu, softmax	13,25	13,61	0,74765	160,59
640, 128, 42	tanh, sigmoide, softmax	12,49	12,97	0,83135	166,96
640, 128, 42	tanh, tanh, softmax	12,39	13,01	0,84840	164,43
640, 128, 42	tanh, relu, softmax	12,62	13,19	0,82100	164,58
640, 256, 42	tanh, sigmoide, softmax	12,02	12,55	0,89070	179,04
640, 256, 42	tanh, tanh, softmax	11,97	12,65	0,89250	178,04
640, 256, 42	tanh, relu, softmax	12,25	12,85	0,87695	177,68
640, 64, 42	relu, sigmoide, softmax	11,91	12,49	0,89980	163,31
640, 64, 42	relu, tanh, softmax	11,82	12,51	0,91890	164,35
640, 64, 42	relu, relu, softmax	11,89	12,53	0,91340	164,91
640, 128, 42	relu, sigmoide, softmax	11,78	12,41	0,91435	177,77
640, 128, 42	relu, tanh, softmax	11,79	12,52	0,91990	186,78
640, 128, 42	relu, relu, softmax	11,84	12,51	0,92145	182,40
640, 256, 42	relu, sigmoide, softmax	11,75	12,38	0,92415	194,53
640, 256, 42	relu, tanh, softmax	11,83	12,52	0,92115	183,18
640, 256, 42	relu, relu, softmax	11,76	12,36	**0,92720**	195,45

Fonte: O autor (2019).

O melhor resultado, representado pela maior acurácia, entre as combinações de topologias com 640 neurônios de entrada, para a técnica ANN, foi a topologia ilustrada na FIGURA 42, pela qual foi obtida uma acurácia de 0,9272, representando 18.544 circuitos dos 20.000 contidos no conjunto de testes determinados corretamente.

FIGURA 42 – MELHOR RESULTADO ANN - CIRCUITOS 3X2 E 640 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 17 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 3 q-bits com 2 operações quânticas, com 1.280 neurônios na primeira camada.

TABELA 17 – RESULTADOS PARA A ANN - CIRCUITOS 3X2 E 1280 NEURÔNIOS DE ENTRADA

(continua)

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
1280, 64, 42	sigmoide, sigmoide, softmax	14,34	14,51	0,62305	184,15
1280, 64, 42	sigmoide, tanh, softmax	15,07	15,17	0,54650	188,34
1280, 64, 42	sigmoide, relu, softmax	15,93	15,94	0,40955	203,46
1280, 128, 42	sigmoide, sigmoide, softmax	13,54	13,75	0,71795	222,35
1280, 128, 42	sigmoide, tanh, softmax	14,15	14,36	0,64745	220,29
1280, 128, 42	sigmoide, relu, softmax	15,06	15,13	0,53855	217,00
1280, 256, 42	sigmoide, sigmoide, softmax	12,94	13,18	0,79335	247,16
1280, 256, 42	sigmoide, tanh, softmax	13,66	13,92	0,69595	251,84
1280, 256, 42	sigmoide, relu, softmax	14,13	14,24	0,65645	256,72
1280, 64, 42	tanh, sigmoide, softmax	13,31	13,68	0,73945	183,91
1280, 64, 42	tanh, tanh, softmax	13,42	13,86	0,72225	185,79
1280, 64, 42	tanh, relu, softmax	13,46	13,80	0,73865	179,99
1280, 128, 42	tanh, sigmoide, softmax	12,53	13,02	0,82520	198,33
1280, 128, 42	tanh, tanh, softmax	12,54	13,14	0,82035	195,52
1280, 128, 42	tanh, relu, softmax	12,74	13,26	0,81110	197,23
1280, 256, 42	tanh, sigmoide, softmax	12,01	12,61	0,88465	296,86
1280, 256, 42	tanh, tanh, softmax	12,07	12,74	0,88410	303,88
1280, 256, 42	tanh, relu, softmax	12,33	12,97	0,86525	307,43
1280, 64, 42	relu, sigmoide, softmax	11,80	12,43	0,91640	220,98
1280, 64, 42	relu, tanh, softmax	11,78	12,49	0,91585	211,41

TABELA 17 – RESULTADOS PARA A ANN - CIRCUITOS 3X2 E 1280 NEURÔNIOS DE ENTRADA

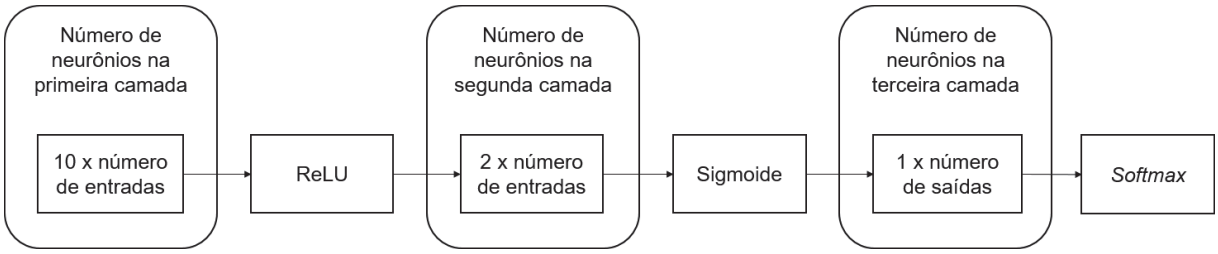
(conclusão)

Número de neurônios (1ª, 2ª, 3ª camada)	Funções de ativação (1ª, 2ª, 3ª camada)	MSE (treinamento)	MSE (validação)	Acurácia	Tempo de treinamento (s)
1280, 64, 42	relu, relu, softmax	11,88	12,48	0,92040	196,74
1280, 128, 42	relu, sigmoide, softmax	11,68	12,34	0,92810	199,91
1280, 128, 42	relu, tanh, softmax	11,72	12,39	0,92770	208,57
1280, 128, 42	relu, relu, softmax	11,80	12,40	0,92935	261,26
1280, 256, 42	relu, sigmoide, softmax	11,66	12,30	**0,93445**	374,60
1280, 256, 42	relu, tanh, softmax	11,74	12,33	0,93405	402,34
1280, 256, 42	relu, relu, softmax	11,73	12,33	0,93095	411,05

Fonte: O autor (2019).

A topologia que apresentou o melhor resultado, entre as combinações com 1.280 neurônios de entrada, foi a topologia apresentada na FIGURA 43. Para esta topologia, a acurácia foi de 0,93445, ou seja, 18.689 circuitos dos 20.000 contidos no conjunto de testes foram determinados corretamente.

FIGURA 43 – MELHOR RESULTADO ANN - CIRCUITOS 3X2 E 1280 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

Comparando-se os resultados obtidos nos quatro testes, o melhor resultado para a compilação de algoritmos quânticos, de 3 q-bits com 2 operações quânticas, foi obtido pela topologia com 1.280 neurônios na primeira camada, 256 neurônios na segunda camada e 42 neurônios na camada de saída, com funções de ativação RELU, sigmoide e *softmax*, respectivamente, com uma acurácia de 0,93445, ou seja, 18.689 circuitos dos 20.000 contidos no conjunto de testes foram determinados corretamente.

5.1.4 Resultados para circuitos 3x3

A TABELA 18 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 3 q-bits com 3 operações quânticas, com 128 neurônios na primeira camada.

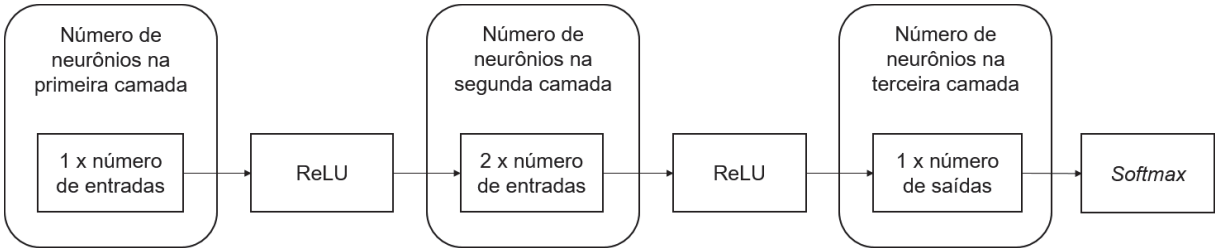
TABELA 18 – RESULTADOS PARA A ANN - CIRCUITOS 3X3 E 128 NEURÔNIOS DE ENTRADA

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
128, 64, 63	sigmoide, sigmoide, softmax	28,57	28,78	0,28290	107,95
128, 64, 63	sigmoide, tanh, softmax	28,56	28,82	0,30245	96,09
128, 64, 63	sigmoide, relu, softmax	28,97	29,20	0,25455	95,46
128, 128, 63	sigmoide, sigmoide, softmax	28,21	28,47	0,35165	97,83
128, 128, 63	sigmoide, tanh, softmax	28,35	28,65	0,33785	109,53
128, 128, 63	sigmoide, relu, softmax	28,49	28,80	0,31690	113,05
128, 256, 63	sigmoide, sigmoide, softmax	27,94	28,24	0,37320	117,70
128, 256, 63	sigmoide, tanh, softmax	28,48	28,81	0,28690	131,98
128, 256, 63	sigmoide, relu, softmax	27,88	28,35	0,36725	133,22
128, 63	sigmoide, softmax	29,40	29,66	0,24510	128,78
128, 64, 63	tanh, sigmoide, softmax	27,64	28,14	0,39710	126,21
128, 64, 63	tanh, tanh, softmax	27,62	28,34	0,40785	149,34
128, 64, 63	tanh, relu, softmax	27,61	28,32	0,39395	157,22
128, 128, 63	tanh, sigmoide, softmax	27,11	27,76	0,44390	165,22
128, 128, 63	tanh, tanh, softmax	26,86	27,95	0,45845	171,96
128, 128, 63	tanh, relu, softmax	26,93	28,10	0,44395	168,09
128, 256, 63	tanh, sigmoide, softmax	26,65	27,48	0,47270	179,19
128, 256, 63	tanh, tanh, softmax	26,27	27,61	0,48475	179,36
128, 256, 63	tanh, relu, softmax	26,05	27,93	0,49055	183,50
128, 63	tanh, softmax	28,40	28,83	0,31475	166,94
128, 64, 63	relu, sigmoide, softmax	27,21	27,97	0,40735	158,22
128, 64, 63	relu, tanh, softmax	27,03	28,12	0,42730	120,39
128, 64, 63	relu, relu, softmax	26,81	27,95	0,45090	109,86
128, 128, 63	relu, sigmoide, softmax	26,88	27,75	0,43895	107,33
128, 128, 63	relu, tanh, softmax	26,87	28,10	0,44225	109,40
128, 128, 63	relu, relu, softmax	26,24	27,73	0,48875	108,56
128, 256, 63	relu, sigmoide, softmax	26,53	27,54	0,47290	114,41
128, 256, 63	relu, tanh, softmax	27,08	28,29	0,41555	125,16
128, 256, 63	relu, relu, softmax	25,50	27,61	**0,52320**	128,17
128, 63	relu, softmax	27,45	28,38	0,38820	118,89

Fonte: O autor (2019).

Dentre as combinações com 128 neurônios de entrada, a topologia que apresentou o melhor resultado foi a apresentada na FIGURA 44, que obteve uma acurácia foi de 0,5232, representando 10.464 circuitos determinados corretamente dos 20.000 contidos no conjunto de testes.

FIGURA 44 – MELHOR RESULTADO ANN - CIRCUITOS 3X3 E 128 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 19 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 3 q-bits com 3 operações quânticas, com 256 neurônios na primeira camada.

TABELA 19 – RESULTADOS PARA A ANN - CIRCUITOS 3X3 E 256 NEURÔNIOS DE ENTRADA

(continua)

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
256, 64, 63	sigmoide, sigmoide, softmax	28,28	28,55	0,34225	123,94
256, 64, 63	sigmoide, tanh, softmax	28,39	28,71	0,33350	125,76
256, 64, 63	sigmoide, relu, softmax	29,57	29,70	0,14785	124,53
256, 128, 63	sigmoide, sigmoide, softmax	27,89	28,22	0,37945	129,15
256, 128, 63	sigmoide, tanh, softmax	27,97	28,37	0,36890	129,60
256, 128, 63	sigmoide, relu, softmax	28,62	28,91	0,28440	132,88
256, 256, 63	sigmoide, sigmoide, softmax	27,49	27,93	0,39235	144,69
256, 256, 63	sigmoide, tanh, softmax	27,74	28,21	0,35905	141,26
256, 256, 63	sigmoide, relu, softmax	28,03	28,49	0,36265	146,32
256, 64, 63	tanh, sigmoide, softmax	27,47	28,11	0,41455	136,41
256, 64, 63	tanh, tanh, softmax	27,50	28,36	0,41115	139,87
256, 64, 63	tanh, relu, softmax	27,56	28,31	0,38915	137,21
256, 128, 63	tanh, sigmoide, softmax	26,75	27,64	0,46705	143,92
256, 128, 63	tanh, tanh, softmax	26,52	27,90	0,47255	142,79
256, 128, 63	tanh, relu, softmax	26,80	28,08	0,44190	139,20
256, 256, 63	tanh, sigmoide, softmax	26,03	27,24	0,50980	148,09

TABELA 19 – RESULTADOS PARA A ANN - CIRCUITOS 3X3 E 256 NEURÔNIOS DE ENTRADA

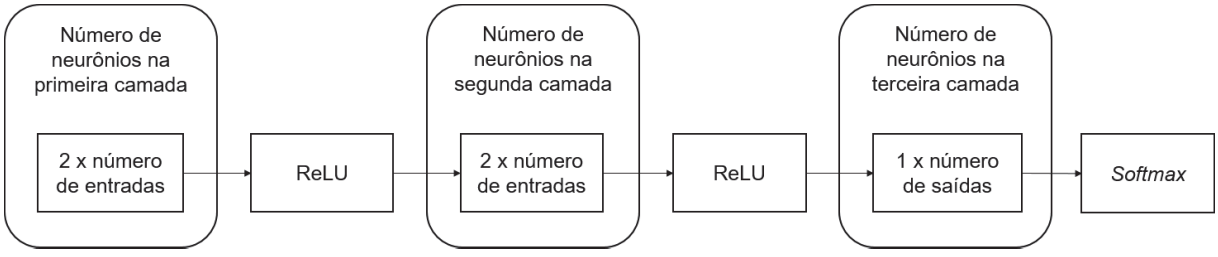
(conclusão)

Número de neurônios (1ª, 2ª, 3ª camada)	Funções de ativação (1ª, 2ª, 3ª camada)	MSE (treinamento)	MSE (validação)	Acurácia	Tempo de treinamento (s)
256, 256, 63	tanh, tanh, softmax	25,61	27,63	0,51755	151,30
256, 256, 63	tanh, relu, softmax	25,82	28,05	0,50255	155,78
256, 64, 63	relu, sigmoide, softmax	26,41	27,62	0,46525	143,61
256, 64, 63	relu, tanh, softmax	26,02	27,91	0,48645	142,37
256, 64, 63	relu, relu, softmax	25,90	27,76	0,48700	143,49
256, 128, 63	relu, sigmoide, softmax	25,96	27,41	0,49080	147,65
256, 128, 63	relu, tanh, softmax	25,80	27,95	0,49355	148,75
256, 128, 63	relu, relu, softmax	25,30	27,67	0,52575	152,21
256, 256, 63	relu, sigmoide, softmax	25,62	27,28	0,50800	157,28
256, 256, 63	relu, tanh, softmax	25,86	28,10	0,50285	155,73
256, 256, 63	relu, relu, softmax	24,52	27,76	**0,55630**	162,76

Fonte: O autor (2019).

O melhor resultado, representado pela maior acurácia, entre as combinações de topologias com 256 neurônios de entrada, para a técnica ANN, foi a topologia ilustrada na FIGURA 45, pela qual foi obtida uma acurácia de 0,5563, representando 11.126 circuitos dos 20.000 contidos no conjunto de testes determinados corretamente.

FIGURA 45 – MELHOR RESULTADO ANN - CIRCUITOS 3X3 E 256 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 20 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 3 q-bits com 3 operações quânticas, com 640 neurônios na primeira camada.

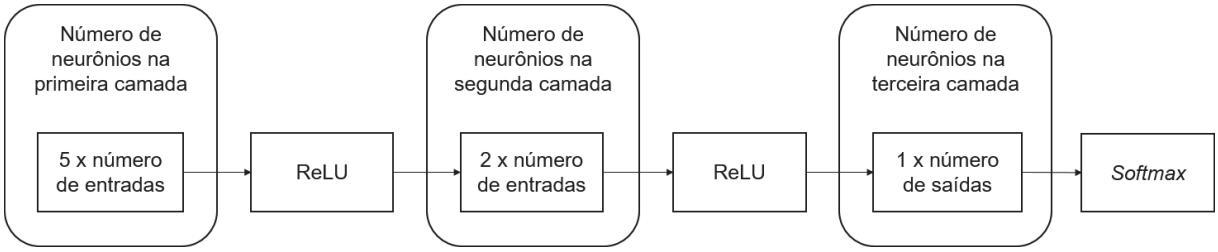
TABELA 20 – RESULTADOS PARA A ANN - CIRCUITOS 3X3 E 640 NEURÔNIOS DE ENTRADA

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
640, 64, 63	sigmoide, sigmoide, softmax	28,16	28,53	0,35610	163,74
640, 64, 63	sigmoide, tanh, softmax	28,85	29,13	0,28330	167,43
640, 64, 63	sigmoide, relu, softmax	29,78	29,91	0,17070	161,25
640, 128, 63	sigmoide, sigmoide, softmax	27,58	28,04	0,39700	159,21
640, 128, 63	sigmoide, tanh, softmax	28,23	28,65	0,34810	160,18
640, 128, 63	sigmoide, relu, softmax	28,98	29,21	0,28125	161,39
640, 256, 63	sigmoide, sigmoide, softmax	27,04	27,64	0,43445	175,62
640, 256, 63	sigmoide, tanh, softmax	27,70	28,20	0,39570	172,41
640, 256, 63	sigmoide, relu, softmax	28,49	28,81	0,31550	176,12
640, 64, 63	tanh, sigmoide, softmax	27,46	28,23	0,40530	157,69
640, 64, 63	tanh, tanh, softmax	27,57	28,45	0,40595	175,72
640, 64, 63	tanh, relu, softmax	27,74	28,42	0,38685	197,44
640, 128, 63	tanh, sigmoide, softmax	26,52	27,68	0,47975	201,88
640, 128, 63	tanh, tanh, softmax	26,51	28,01	0,47460	193,62
640, 128, 63	tanh, relu, softmax	26,93	28,23	0,43490	202,49
640, 256, 63	tanh, sigmoide, softmax	25,55	27,26	0,52875	227,34
640, 256, 63	tanh, tanh, softmax	25,27	27,74	0,54225	226,15
640, 256, 63	tanh, relu, softmax	25,86	28,30	0,47990	212,08
640, 64, 63	relu, sigmoide, softmax	25,25	27,39	0,52505	171,81
640, 64, 63	relu, tanh, softmax	24,52	28,05	0,55055	185,34
640, 64, 63	relu, relu, softmax	24,72	27,82	0,55160	190,95
640, 128, 63	relu, sigmoide, softmax	24,69	27,28	0,55820	179,15
640, 128, 63	relu, tanh, softmax	24,17	28,14	0,57160	176,20
640, 128, 63	relu, relu, softmax	24,15	27,84	0,58010	180,00
640, 256, 63	relu, sigmoide, softmax	24,28	27,28	0,58315	195,45
640, 256, 63	relu, tanh, softmax	24,16	28,28	0,56555	190,17
640, 256, 63	relu, relu, softmax	23,45	28,02	**0,60990**	196,02

Fonte: O autor (2019).

A topologia que apresentou o melhor resultado, entre as combinações com 640 neurônios de entrada, foi a topologia apresentada na FIGURA 46. Para esta topologia, a acurácia foi de 0,6099, ou seja, 12.198 circuitos dos 20.000 contidos no conjunto de testes foram determinados corretamente.

FIGURA 46 – MELHOR RESULTADO ANN - CIRCUITOS 3X3 E 640 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

A TABELA 21 apresenta os resultados obtidos para as diferentes topologias propostas de ANN para a compilação de circuitos quânticos de 3 q-bits com 3 operações quânticas, com 1.280 neurônios na primeira camada.

TABELA 21 – RESULTADOS PARA A ANN - CIRCUITOS 3X3 E 1280 NEURÔNIOS DE ENTRADA

(continua)

Número de neurônios (1ª, 2ª, 3ª camada)	**Funções de ativação (1ª, 2ª, 3ª camada)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
1280, 64, 63	sigmoide, sigmoide, softmax	28,85	29,10	0,29265	199,44
1280, 64, 63	sigmoide, tanh, softmax	29,45	29,65	0,23400	193,55
1280, 64, 63	sigmoide, relu, softmax	29,65	29,77	0,22960	195,82
1280, 128, 63	sigmoide, sigmoide, softmax	28,28	28,65	0,32775	245,10
1280, 128, 63	sigmoide, tanh, softmax	29,30	29,55	0,22960	287,97
1280, 128, 63	sigmoide, relu, softmax	29,41	29,57	0,22175	335,12
1280, 256, 63	sigmoide, sigmoide, softmax	27,44	28,00	0,40210	441,45
1280, 256, 63	sigmoide, tanh, softmax	28,66	28,99	0,31940	444,22
1280, 256, 63	sigmoide, relu, softmax	28,44	28,73	0,31845	444,61
1280, 64, 63	tanh, sigmoide, softmax	27,59	28,35	0,39530	353,37
1280, 64, 63	tanh, tanh, softmax	27,76	28,58	0,40100	354,76
1280, 64, 63	tanh, relu, softmax	27,87	28,50	0,35130	323,13
1280, 128, 63	tanh, sigmoide, softmax	26,57	27,80	0,46940	231,71
1280, 128, 63	tanh, tanh, softmax	26,60	28,02	0,46915	232,95
1280, 128, 63	tanh, relu, softmax	27,01	28,25	0,42840	285,91
1280, 256, 63	tanh, sigmoide, softmax	25,50	27,37	0,52750	381,26
1280, 256, 63	tanh, tanh, softmax	25,44	27,79	0,52715	376,09
1280, 256, 63	tanh, relu, softmax	25,98	28,38	0,48270	375,69
1280, 64, 63	relu, sigmoide, softmax	24,38	27,38	0,55970	275,73
1280, 64, 63	relu, tanh, softmax	23,57	28,17	0,59595	297,48

TABELA 21 – RESULTADOS PARA A ANN - CIRCUITOS 3X3 E 1280 NEURÔNIOS DE ENTRADA

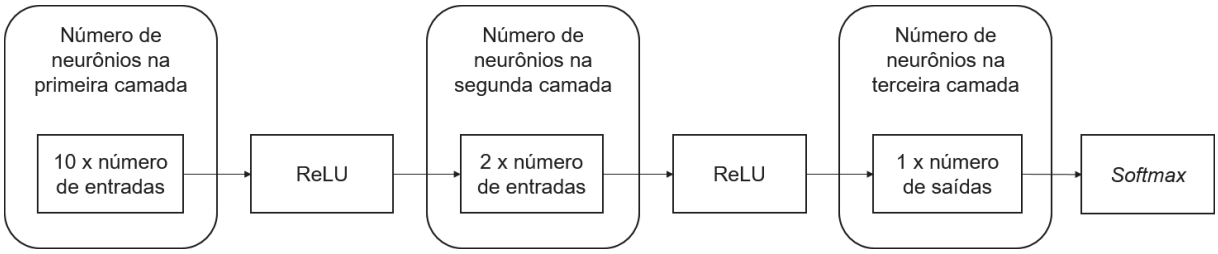
(conclusão)

Número de neurônios (1ª, 2ª, 3ª camada)	Funções de ativação (1ª, 2ª, 3ª camada)	MSE (treinamento)	MSE (validação)	Acurácia	Tempo de treinamento (s)
1280, 64, 63	relu, relu, softmax	24,11	27,97	0,57400	286,07
1280, 128, 63	relu, sigmoide, softmax	23,75	27,41	0,59420	338,78
1280, 128, 63	relu, tanh, softmax	23,18	28,36	0,61035	344,48
1280, 128, 63	relu, relu, softmax	23,45	28,05	0,60160	326,76
1280, 256, 63	relu, sigmoide, softmax	23,29	27,47	0,61720	400,46
1280, 256, 63	relu, tanh, softmax	23,14	28,59	0,61140	340,00
1280, 256, 63	relu, relu, softmax	22,98	28,22	**0,62545**	334,76

Fonte: O autor (2019).

Considerando as combinações de topologias de ANN, com 1 280 neurônios de entrada, a que apresentou o melhor resultado foi a topologia apresentada na FIGURA 47, obtendo 12.509 circuitos dos 20.000 contidos no conjunto de testes corretamente, resultado representado por uma acurácia de 0,62545.

FIGURA 47 – MELHOR RESULTADO ANN - CIRCUITOS 3X3 E 1280 NEURÔNIOS DE ENTRADA



FONTE: O autor (2019).

Comparando-se os resultados obtidos nos quatro testes, o melhor resultado para a compilação de algoritmos quânticos, de 3 q-bits com 3 operações quânticas, foi obtido pela topologia com 1.280 neurônios na primeira camada, 256 neurônios na segunda camada e 63 neurônios na camada de saída, com funções de ativação RELU, RELU e *softmax*, respectivamente, com uma acurácia de 0,62545, ou seja, 12.509 circuitos dos 20.000 contidos no conjunto de testes foram determinados corretamente.

5.2 REDE NEURAL CONVOLUCIONAL

Os itens seguintes apresentam os resultados para as quatro topologias de circuitos quânticos apresentadas, e as tabelas contidas nesses itens estão divididas entre as combinações de diferentes tamanhos de *kernel* apresentadas na FIGURA 29, que são ¼ x ¼, ¼ x ½, ½ x ¼ e ½ x ½ das dimensões das entradas propriamente ditas, ou seja, para um circuito com 2 q-bits, onde o resultado do circuito é sempre uma matriz complexa de 4 x 4, as dimensões de entrada serão 8 x 4, pois, como apresentado no item 4.3, a parte real e imaginária da matriz é separada.

5.2.1 Resultados para circuitos 2x2

A ANN escolhida para compor a CNN desse teste foi a que obteve o melhor resultado no item 5.1.1, que foi obtido pela topologia com 160 neurônios na primeira camada, 64 neurônios na segunda camada e 28 neurônios na camada de saída, com funções de ativação RELU, tangente hiperbólica e *softmax*, respectivamente.

A TABELA 22 apresenta os resultados obtidos para as diferentes dimensões de *kernel* da CNN para a compilação de circuitos quânticos de 2 q-bits com 2 operações quânticas.

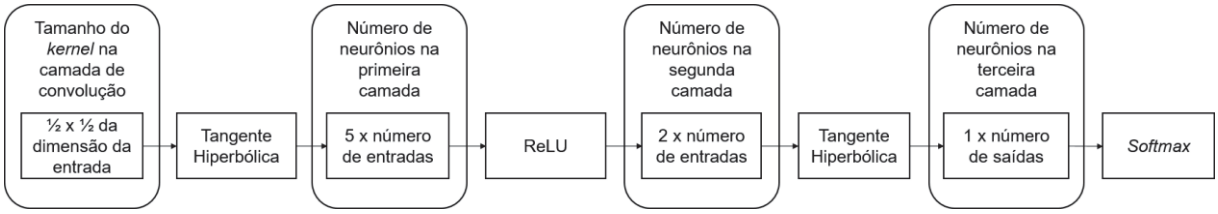
TABELA 22 – RESULTADOS PARA A CNN - CIRCUITOS 2X2

Dimensões de *kernel* (linhas, colunas)	**Função de ativação (*kernel*)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
2, 1	sigmoide	6,83	6,88	0,86780	196,02
2, 1	tanh	6,69	6,88	0,86720	193,86
2, 1	relu	6,77	6,79	0,88560	198,11
2, 2	sigmoide	6,67	6,78	0,88670	191,95
2, 2	tanh	6,82	6,88	0,86750	195,17
2, 2	relu	6,81	6,85	0,87380	194,89
4, 1	sigmoide	6,77	6,82	0,87870	196,01
4, 1	tanh	6,77	6,91	0,86070	196,00
4, 1	relu	6,65	6,88	0,86710	193,64
4, 2	sigmoide	6,80	6,81	0,88090	195,31
4, 2	tanh	6,66	6,72	**0,89950**	195,04
4, 2	relu	6,65	6,72	0,89870	196,63

Fonte: O autor (2019).

A FIGURA 48 apresenta a topologia de CNN com a dimensão de *kernel* que obteve o melhor resultado.

FIGURA 48 – MELHOR RESULTADO CNN - CIRCUITOS 2X2



FONTE: O autor (2019).

Para esta topologia, a acurácia foi de 0,8995, ou seja, 17.990 circuitos dos 20.000 contidos no conjunto de testes foram determinados corretamente.

5.2.2 Resultados para circuitos 2x3

A ANN escolhida para compor a CNN desse teste foi a que obteve o melhor resultado no item 5.1.2, que foi obtido pela topologia com 320 neurônios na primeira camada, 64 neurônios na segunda camada e 42 neurônios na camada de saída, com funções de ativação RELU, tangente hiperbólica e *softmax*, respectivamente.

A TABELA 23 apresenta os resultados obtidos para as diferentes dimensões de *kernel* da CNN para a compilação de circuitos quânticos de 2 q-bits com 3 operações quânticas.

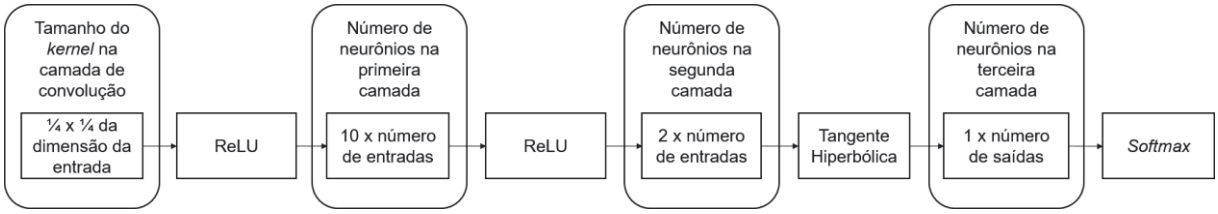
TABELA 23 – RESULTADOS PARA A CNN - CIRCUITOS 2X3

Dimensões de *kernel* (linhas, colunas)	**Função de ativação (*kernel*)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
2, 1	sigmoide	13,62	13,85	0,75680	239,45
2, 1	tanh	13,57	13,91	0,74520	243,95
2, 1	relu	13,64	13,75	**0,77650**	244,37
2, 2	sigmoide	13,65	13,83	0,76030	241,13
2, 2	tanh	13,66	13,80	0,76800	242,51
2, 2	relu	13,59	13,84	0,75910	238,36
4, 1	sigmoide	13,64	13,89	0,74880	238,42
4, 1	tanh	13,63	13,90	0,74700	242,17
4, 1	relu	13,66	13,81	0,76540	244,03
4, 2	sigmoide	13,65	13,84	0,76000	249,62
4, 2	tanh	13,66	13,87	0,75390	239,86
4, 2	relu	13,61	13,85	0,75820	242,42

Fonte: O autor (2019).

A topologia de CNN com a dimensão de *kernel* que obteve o melhor resultado está ilustrada pela FIGURA 49, pois obteve acurácia de 0,7765, representando 15.530 circuitos dos 20.000 contidos no conjunto de testes determinados corretamente.

FIGURA 49 – MELHOR RESULTADO CNN - CIRCUITOS 2X3



FONTE: O autor (2019).

5.2.3 Resultados para circuitos 3x2

A ANN escolhida para compor a CNN desse teste foi a que obteve o melhor resultado no item 5.1.3, que foi obtido pela topologia com 1.280 neurônios na primeira camada, 256 neurônios na segunda camada e 42 neurônios na camada de saída, com funções de ativação RELU, sigmoide e *softmax*, respectivamente.

A TABELA 24 apresenta os resultados obtidos para as diferentes dimensões de *kernel* da CNN para a compilação de circuitos quânticos de 3 q-bits com 2 operações quânticas.

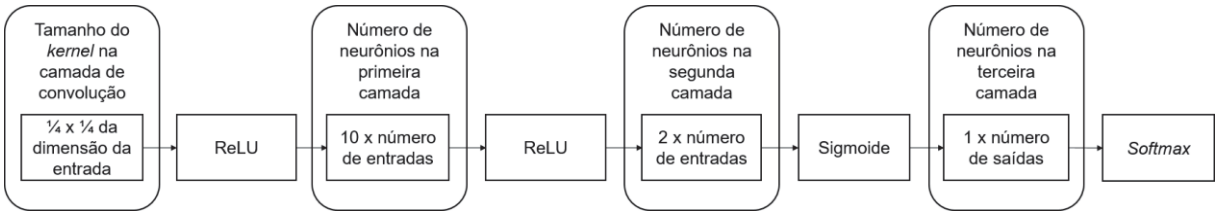
TABELA 24 – RESULTADOS PARA A CNN - CIRCUITOS 3x2

Dimensões de *kernel* (linhas, colunas)	**Função de ativação (*kernel*)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
4, 2	sigmoide	12,96	13,37	0,74680	562,70
4, 2	tanh	12,96	13,32	0,75720	551,59
4, 2	relu	12,89	13,14	**0,79350**	560,56
4, 4	sigmoide	12,93	13,28	0,76410	552,42
4, 4	tanh	12,83	13,19	0,78250	568,23
4, 4	relu	13,00	13,25	0,77110	557,86
8, 2	sigmoide	12,91	13,27	0,76690	563,79
8, 2	tanh	12,88	13,30	0,76110	559,92
8, 2	relu	12,93	13,25	0,77130	565,01
8, 4	sigmoide	12,87	13,34	0,75280	559,59
8, 4	tanh	12,94	13,24	0,77340	566,13
8, 4	relu	12,91	13,28	0,76530	567,63

Fonte: O autor (2019).

Dentre as diferentes configurações de tamanhos de *kernel* testados, A topologia de CNN apresentada na FIGURA 50 foi a que obteve a maior acurácia, determinando corretamente 15.870 circuitos dos 20.000 contidos no conjunto de testes, representando uma acurácia de 0,7935.

FIGURA 50 – MELHOR RESULTADO CNN - CIRCUITOS 3x2



FONTE: O autor (2019).

5.2.4 Resultados para circuitos 3x3

A ANN escolhida para compor a CNN desse teste foi a que obteve o melhor resultado no item 5.1.4, que foi obtido pela topologia com 1.280 neurônios na primeira camada, 256 neurônios na segunda camada e 63 neurônios na camada de saída, com funções de ativação RELU, RELU e *softmax*, respectivamente.

A TABELA 25 apresenta os resultados obtidos para as diferentes dimensões de *kernel* da CNN para a compilação de circuitos quânticos de 3 q-bits com 3 operações quânticas.

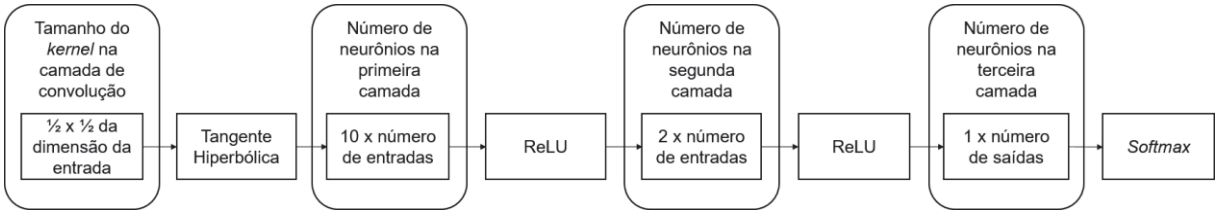
TABELA 25 – RESULTADOS PARA A CNN - CIRCUITOS 3x3

Dimensões de *kernel* (linhas, colunas)	**Função de ativação (*kernel*)**	**MSE (treinamento)**	**MSE (validação)**	**Acurácia**	**Tempo de treinamento (s)**
4, 2	sigmoide	25,47	27,57	0,59790	588,37
4, 2	tanh	25,44	27,58	0,59440	576,93
4, 2	relu	25,48	27,66	0,57840	565,27
4, 4	sigmoide	25,46	27,55	0,60180	573,06
4, 4	tanh	25,40	27,63	0,58540	586,23
4, 4	relu	25,35	27,61	0,59010	571,67
8, 2	sigmoide	25,53	27,58	0,59550	583,86
8, 2	tanh	25,41	27,54	0,60380	578,84
8, 2	relu	25,45	27,58	0,59600	586,72
8, 4	sigmoide	25,47	27,59	0,59410	566,33
8, 4	tanh	25,45	27,52	**0,60790**	576,91
8, 4	relu	25,39	27,60	0,59130	570,85

Fonte: O autor (2019).

Por fim, a FIGURA 51 traz a topologia de CNN com a dimensão de *kernel* que determinou corretamente 12.158 circuitos dos 20.000 contidos no conjunto de testes, representando o melhor resultado com uma acurácia de 0,6079.

FIGURA 51 – MELHOR RESULTADO CNN - CIRCUITOS 3x3



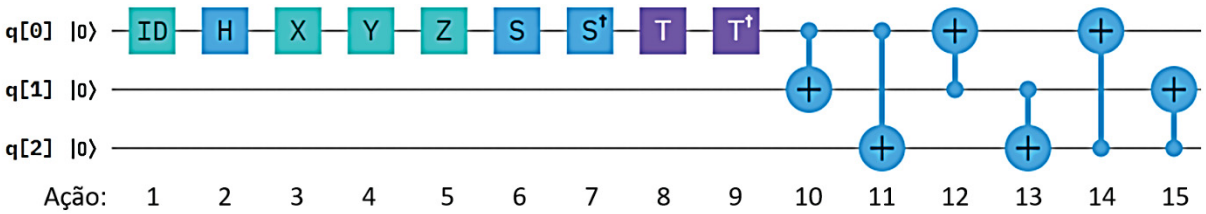
FONTE: O autor (2019).

5.3 DEEP Q-NETWORK

Diferentemente dos testes executados para a ANN e para a CNN, onde a compilação é resultado de uma rede já treinada, para a DQN é necessário um treinamento para cada circuito que se deseja compilar, ou seja, o custo computacional para a compilação de cada circuito da DQN é superior se comparado às duas técnicas anteriores, exigindo um conjunto de circuitos limitado para fins de teste. Sendo assim, um conjunto de 10 circuitos diferentes foram escolhidos para cada combinação de q-bits e operações quânticas, resultando em 40 circuitos distintos, gerados aleatoriamente. Todos os testes foram realizados utilizando 5 000 episódios, por demandar um tempo de execução plausível para os testes realizados.

Devido à natureza do problema, o número de neurônios na última camada de cada ANN, utilizada para regredir a função Q que é apresentada no item 3.1.3.3, não pôde ser mantido, pois a camada de saída, para a DQN, representa o número de ações possíveis para a quantidade de q-bits no circuito a ser compilado. Considerando um circuito de 3 q-bits, por exemplo, são possíveis 15 ações, ou seja, 9 portas quânticas de 1 q-bit e 6 combinações de CNOT que operam 2 q-bits ao mesmo tempo, como mostrado no item 4.4. Sendo assim, as ANNs utilizadas para os testes da DQN não possuem a mesma quantidade de neurônios na última camada que nas versões com melhor resultado, apresentadas no item 5.1. A FIGURA 52 apresenta as possíveis ações para um circuito de 3 q-bits.

FIGURA 52 – CONJUNTO DE AÇÕES PARA A DQN – 3 Q-BITS



FONTE: O autor (2019).

Como definido no item 4.6.3, somente o desempenho de dois parâmetros da técnica DQN foram avaliados. As tabelas comparativas dos itens a seguir apresentam as combinações entre os fatores de desconto de recompensa e os fatores de decaimentos da exploração, assim como a acurácia e o tempo de treinamento para cada combinação. Devido a diferença entre as técnicas ANN e CNN em relação a DQN, os valores de erro médio quadrático não foram adicionados a tabela por não trazerem nenhuma informação relevante para a comparação de desempenho, já que essa métrica possui um significado diferente na DQN, como apresentado no item 3.1.3.4.

5.3.1 Resultados para circuitos 2x2

A ANN escolhida para regredir a função Q, apresentada no item 3.1.3.3, foi a que obteve o melhor resultado no item 5.1.1, sendo ela a topologia com 160 neurônios na primeira camada e 64 neurônios na segunda e 11 neurônios na camada de saída, com funções de ativação RELU, tangente hiperbólica e *softmax*, respectivamente. TABELA 26 apresenta os resultados obtidos para as combinações de fatores de desconto de recompensa e os fatores de decaimentos da exploração, para a compilação de circuitos quânticos de 2 q-bits com 2 operações quânticas.

TABELA 26 – RESULTADOS PARA A DQN - CIRCUITOS 2X2

Fator de desconto de recompensa	Fator de decaimento da exploração	Acurácia	Tempo de treinamento (s)
0,990	0,999	**1,00000**	463,57
0,990	0,990	**1,00000**	471,35
0,990	0,950	**1,00000**	446,79
0,950	0,999	**1,00000**	465,52
0,950	0,990	**1,00000**	462,25
0,950	0,950	**1,00000**	452,50
0,900	0,999	**1,00000**	457,74
0,900	0,990	**1,00000**	462,40
0,900	0,950	**1,00000**	481,81
0,800	0,999	**1,00000**	476,96
0,800	0,990	**1,00000**	**452,24**
0,800	0,950	**1,00000**	478,55

Fonte: O autor (2019).

As combinações de fatores escolhidos não apresentaram influência na acurácia da técnica para o problema da compilação quântica, já que a DQN foi capaz de encontrar a resposta correta para os 10 circuitos quânticos de testes arbitrados para todas as combinações de fatores de decaimento da exploração e de recompensa escolhidos.

5.3.2 Resultados para circuitos 2x3

A ANN escolhida para regredir a função Q, apresentada no item 3.1.3.3, foi a que obteve o melhor resultado no item 5.1.2, sendo ela a topologia com 320 neurônios na primeira camada e 64 neurônios na segunda e 11 neurônios na camada de saída, com funções de ativação RELU, tangente hiperbólica e *softmax*, respectivamente.

A TABELA 27 apresenta os resultados obtidos para as combinações de fatores de desconto de recompensa e os fatores de decaimentos da exploração, para a compilação de circuitos quânticos de 2 q-bits com 3 operações quânticas.

TABELA 27 – RESULTADOS PARA A DQN - CIRCUITOS 2X3

Fator de desconto de recompensa	Fator de decaimento da exploração	Acurácia	Tempo de treinamento (s)
0,990	0,999	**0,70000**	793,26
0,990	0,990	**0,70000**	786,95
0,990	0,950	**0,70000**	793,17
0,950	0,999	**0,70000**	785,82
0,950	0,990	**0,70000**	786,46
0,950	0,950	**0,70000**	799,37
0,900	0,999	**0,70000**	798,73
0,900	0,990	**0,70000**	798,79
0,900	0,950	**0,70000**	792,83
0,800	0,999	**0,70000**	**777,80**
0,800	0,990	**0,70000**	793,19
0,800	0,950	**0,70000**	800,50

Fonte: O autor (2019).

Para todas as combinações de fatores escolhidos, a DQN obteve uma acurácia de 0,7, ou seja, o método foi capaz de compilar corretamente 7 dos 10 circuitos quânticos de testes arbitrados, não apresentando influências dos fatores de fatores de decaimento da exploração e de recompensa escolhidos na acurácia.

5.3.3 Resultados para circuitos 3x2

A ANN escolhida para regredir a função Q, apresentada no item 3.1.3.3, foi a que obteve o melhor resultado no item 5.1.3, sendo ela a topologia com 1 280 neurônios na primeira camada e 256 neurônios na segunda e 15 neurônios na camada de saída, com funções de ativação RELU, sigmoide e *softmax*, respectivamente.

A TABELA 28 apresenta os resultados obtidos para as combinações de fatores de desconto de recompensa e os fatores de decaimentos da exploração, para a compilação de circuitos quânticos de 3 q-bits com 2 operações quânticas.

TABELA 28 – RESULTADOS PARA A DQN - CIRCUITOS 3X2

Fator de desconto de recompensa	Fator de decaimento da exploração	Acurácia	Tempo de treinamento (s)
0,990	0,999	**0,80000**	2544,95
0,990	0,990	**0,80000**	2512,54
0,990	0,950	**0,80000**	2576,30
0,950	0,999	**0,80000**	**2483,85**
0,950	0,990	**0,80000**	2532,12
0,950	0,950	**0,80000**	2527,95
0,900	0,999	**0,80000**	2544,77
0,900	0,990	**0,80000**	2544,58
0,900	0,950	**0,80000**	2509,25
0,800	0,999	**0,80000**	2534,29
0,800	0,990	**0,80000**	2530,25
0,800	0,950	**0,80000**	2554,02

Fonte: O autor (2019).

A DQN obteve uma acurácia de 0,8, sendo capaz de compilar corretamente 8 dos 10 circuitos quânticos de testes arbitrados. Novamente, os fatores de decaimento da exploração e de recompensa escolhidos, para o problema e o modelo de compilação quântica proposto, não apresentaram influências na acurácia.

5.3.4 Resultados para circuitos 3x3

A ANN escolhida para regredir a função Q, apresentada no item 3.1.3.3, foi a que obteve o melhor resultado no item 5.1.4, sendo ela a topologia com 1 280 neurônios na primeira camada e 256 neurônios na segunda e 15 neurônios na camada de saída, com funções de ativação RELU, RELU e *softmax*, respectivamente.

A TABELA 29 apresenta os resultados obtidos para as combinações de fatores de desconto de recompensa e os fatores de decaimentos da exploração, para a compilação de circuitos quânticos de 3 q-bits com 3 operações quânticas.

TABELA 29 – RESULTADOS PARA A DQN - CIRCUITOS 3X3

Fator de desconto de recompensa	Fator de decaimento da exploração	Acurácia	Tempo de treinamento (s)
0,990	0,999	**0,50000**	4546,00
0,990	0,990	**0,50000**	4546,80
0,990	0,950	**0,50000**	4448,15
0,950	0,999	**0,50000**	4495,20
0,950	0,990	**0,50000**	**4430,63**
0,950	0,950	**0,50000**	4435,66
0,900	0,999	**0,50000**	4490,99
0,900	0,990	**0,50000**	4567,97
0,900	0,950	**0,50000**	4452,85
0,800	0,999	**0,50000**	4509,90
0,800	0,990	**0,50000**	4480,06
0,800	0,950	**0,50000**	4547,20

Fonte: O autor (2019).

Por fim, obteve-se uma acurácia de 0,5 para a técnica DQN aplicada à compilação de circuitos quânticos de 3 q-bits e 3 operações, ou seja, o método foi capaz de compilar corretamente 5 dos 10 circuitos quânticos de testes arbitrados. Assim como nas três topologias de circuitos quânticos apresentados anteriormente, os fatores de decaimento da exploração e de recompensa escolhidos não apresentaram influências na acurácia.

6 CONCLUSÃO E PESQUISA FUTURA

Nesta dissertação, foi apresentada uma forma de como aplicar técnicas de Aprendizado Profundo para realizar a tarefa da compilação de algoritmos quânticos, até então não explorada na literatura. Para que fosse possível o treinamento dessas técnicas, ferramentas foram desenvolvidas para a geração de um ambiente de interação para o agente da DQN e para a geração de conjuntos de dados de entrada e saída para a ANN e a CNN.

Baseado no dispositivo quântico IBM-Q 5 Tenerife (IBM, 2019), que é um processador quântico de 5 bits, um simulador foi desenvolvido para que se pudesse definir algumas regras para a geração de circuitos, como a quantidade de q-bits e a quantidade de operações, que não era possível utilizando-se a interface fornecida diretamente pelo simulador do próprio IBM-Q 5 Tenerife. Foi criado, então, uma versão local desse simulador, mas com uma interface diferente, para permitir a sua utilização para as três técnicas de Aprendizado Profundo testadas.

Para as técnicas ANN e CNN, foi desenvolvido um gerador de conjunto de dados que leva em conta a limitação de tais técnicas para problemas que não podem ser representados por funções matemáticas. Sendo assim, essa ferramenta gera circuitos aleatoriamente, de acordo com parâmetros determinados pelo usuário, e os testa, via o simulador local desenvolvido, de forma a verificar se são circuitos válidos e se possuem respostas únicas, forçando o problema a ser solucionável por um modelo de função matemática.

Para a técnica DQN, um ambiente de interação foi desenvolvido para que o agente pudesse tomar ações e receber recompensas de acordo com essas ações, criando, assim, um ambiente de decisão de Markov. Esse ambiente funciona como uma interface entre a DQN e o simulador do IBM-Q Tenerife local, pois utiliza as ações do agente, representadas pela entrada de novas portas quânticas no circuito, e insere esses novos circuitos formados no simulador com o objetivo de obter a resposta do circuito e comparar com o circuito objetivo, gerando recompensas de acordo com a proximidade da resposta.

Para testar a viabilidade da técnica DQN, sem forçar o problema a se tornar um problema solucionável por função matemática, demonstrou-se que a técnica *Q-learning* é capaz de considerar e encontrar mais de uma resposta para o mesmo problema. Sendo a DQN uma variação do *Q-learning* onde a função Q é substituída

por uma técnica de Aprendizado Profundo, provou-se, indiretamente, que a DQN é capaz de compilar algoritmos quânticos mesmo quando há mais de um circuito que soluciona o problema.

Para a combinação de 2 q-bits e 2 operações, o melhor resultado foi obtido pela técnica DQN, onde todos os dez circuitos de teste foram compilados corretamente. A variação dos parâmetros utilizados não apresentou nenhum efeito sobre o resultado final, tanto o fator de desconto de recompensa quanto o fator de decaimento de exploração. Ainda para essa configuração de forma de circuito, o segundo melhor resultado foi apresentado pela técnica ANN, utilizando a topologia com 160 neurônios na primeira camada, 64 neurônios na segunda camada e 28 neurônios na camada de saída, com funções de ativação RELU, tangente hiperbólica e *softmax*. Por fim, o terceiro melhor resultado foi obtido pela técnica CNN, utilizando a mesma topologia da ANN com melhor resultado para a sua porção totalmente conectada e *kernel* com dimensões de 4x2 e tangente hiperbólica como função de ativação.

Para a combinação de 2 q-bits e 3 operações, o melhor resultado foi obtido pela técnica ANN, pela topologia com 320 neurônios na primeira camada, 64 neurônios na segunda camada e 42 neurônios na camada de saída, com funções de ativação RELU, tangente hiperbólica e *softmax*, respectivamente. O segundo melhor resultado foi obtido pela DQN, onde não foi encontrada nenhuma influência dos parâmetros de fator de desconto de recompensa e fator de decaimento de exploração sobre o resultado. Por fim, o terceiro melhor resultado foi obtido pela técnica CNN, utilizando a mesma topologia da ANN com melhor resultado para a sua porção totalmente conectada e *kernel* com dimensões de 2x1 e RELU como função de ativação.

Para a combinação de 3 q-bits e 2 operações, o melhor resultado foi obtido pela técnica ANN, pela topologia com 1 280 neurônios na primeira camada, 256 neurônios na segunda camada e 42 neurônios na camada de saída, com funções de ativação RELU, sigmoide e *softmax*, respectivamente. O segundo melhor resultado foi obtido pela DQN, onde não foi encontrada nenhuma influência dos parâmetros de fator de desconto de recompensa e fator de decaimento de exploração sobre o resultado. Por fim, o terceiro melhor resultado foi obtido pela técnica CNN, utilizando a mesma topologia da ANN com melhor resultado para a sua porção totalmente conectada e *kernel* com dimensões de 4x2 e RELU como função de ativação.

Por fim, para a combinação de 3 q-bits e 3 operações, o melhor resultado foi obtido pela técnica ANN, pela topologia com 1 280 neurônios na primeira camada, 256 neurônios na segunda camada e 63 neurônios na camada de saída, com funções de ativação RELU, RELU e *softmax*, respectivamente. O segundo melhor resultado foi obtido pela DQN, onde não foi encontrada nenhuma influência dos parâmetros de fator de desconto de recompensa e fator de decaimento de exploração sobre o resultado. Por fim, o terceiro melhor resultado foi obtido pela técnica CNN, utilizando a mesma topologia da ANN com melhor resultado para a sua porção totalmente conectada e *kernel* com dimensões de 4x2 e tangente hiperbólica como função de ativação.

Como conclusão geral, a técnica que obteve o melhor desempenho foi a ANN, pois obteve 75% dos melhores resultados. A técnica que obteve o pior desempenho foi a CNN, pois obteve 100% dos piores resultados. A técnica DQN apresentou o melhor resultado somente para circuitos de 2 q-bits por 2 operações quânticas, mas conseguiu encontrar todas as respostas corretas em todos os testes, independente dos parâmetros ajustados, indicando que é possível o uso dessa técnica para a compilação de circuitos quânticos e apresentando repetitividade nos resultados, sendo um método mais estável para a tarefa.

O objetivo geral desta dissertação foi projetar um compilador quântico por meio de técnicas de Aprendizado Profundo em que, dado um determinado algoritmo quântico na sua forma matricial, o compilador encontrasse ao menos um circuito quântico, utilizando somente as portas quânticas disponíveis no dispositivo quântico utilizado, que represente o algoritmo determinado. Embora a acurácia das técnicas ANN, CNN e DQN, aplicadas às topologias de circuitos utilizadas, não tenha sido de 100%, o objetivo geral dessa dissertação foi satisfeito, já que foram projetados diferentes compiladores quânticos que encontram a maioria dos circuitos quânticos testados por meio das três técnicas de Aprendizado Profundo escolhidas. Da mesma forma, os objetivos específicos propostos também foram satisfeitos. Foi realizado um apanhado da literatura sobre compiladores quânticos, desenvolvido um simulador de circuitos quânticos baseado no IBM-Q 5 Tenerife e treinadas três técnicas de Aprendizado Profundo para realizarem a tarefa da compilação quântica, sendo elas a ANN, a CNN e a DQN. Por fim, uma comparação utilizando a acurácia como métrica foi realizada para definir qual técnica obteve o melhor desempenho.

Os resultados obtidos ainda são inconclusivos sobre qual método é o melhor, pois a sua acurácia está diretamente ligada a modelagem adotada para o problema

da compilação quântica aqui utilizado. Por não possuir as limitações que a ANN e a CNN possuem, um refino nos modelos de decisão de Markov utilizados podem elevar a acurácia da DQN, por não possuírem a perda de informação gerada na adequação do problema para ser possível a solução por funções matemáticas. Trabalhos futuros irão abordar diferentes métodos de modelagem do ambiente de interação com o agente, com o objetivo de utilizar as vantagens apresentadas da DQN sobre a ANN e a CNN e melhorar os seus resultados para circuitos maiores que 2 q-bits e 2 operações quânticas. Ao final desse trabalho não foi possível criar um compilador quântico universal com acurácia de 100%, mas demonstrou-se que as técnicas de Aprendizado Profundo possuem a capacidade de solucionar o problema da compilação quântica de maneira factível, e que há muitas melhorias que podem ser aplicadas tanto nos modelos, quanto das técnicas utilizadas.

Pesquisas futuras incluirão diferentes modelagens do processo de decisão de Markov para a utilização da DQN, explorando a influência da sequência de preenchimento das portas quânticas no resultado final e investigando outros métodos de recompensa para o agente dessa técnica. Outras técnicas de Aprendizado Profundo, que não possuam a limitação da ANN e CNN de somente modelar funções matemáticas, também serão avaliadas, com o objetivo de obter acurácias mais elevadas que as obtidas nos resultados dessa dissertação e tornar os compiladores quânticos mais genéricos, fazendo com que obtenham acurácia elevada mesmo em circuitos quânticos com mais operações quânticas e um número maior de q-bits.

REFERÊNCIAS

AMY, M.; MOSCA, M. (2017). **T-count optimization and Reed-Muller codes**. Disponível em: <https://arxiv.org/pdf/1601.07363.pdf>. Acesso em: 15 nov. 2018.

ARAUJO, M. P. M. **Síntese evolucionária de circuitos sequenciais inspirada nos princípios da computação quântica**, Dissertação (Mestrado em Engenharia Eletrônica) - Programa de Pós-Graduação em Engenharia Eletrônica, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, RJ, 2008.

ASSAEL, Y. M.; SHILLINGFORD, B.; WHITESON, S.; DE FREITAS, N. LipNet: end-to-end sentence-level lipreading. In: INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS, 5. 2017, Toulon, France. **Proceedings…** Toulon: OpenReview, 2017. p. 1-13.

BARENCO, A.; BENNETT, C. H.; CLEVE, R.; DIVINCENZO, D. P.; MARGOLUS, N.; SHOR, P.; SLEATOR, T.; SMOLIN, J.; WEINFURTER, H. Elementary gates for quantum computation, **Physical Review A**, v. 52, n. 5, p. 3457-3467, 1995.

CAMPBELL, E. T.; HEYFRON, L. E. An efficient quantum compiler that reduces t count. **Quantum Science and Technology**, v. 4, n. 1, p. 015004, 2018.

DAWSON, C. M.; NIELSEN, M. A. **The Solovay-Kitaev algorithm**, Quantum Information and Computation, v. 6, n. 1, p. 81-95, 2006.

DUCLOS-CIANCI, G.; POULIN, D. Reducing the quantum-computing overhead with complex gate distillation, **Physical Review A**, v. 91, n. 4, p. 42315-42324, 2015.

EREMENKO, K. **Deep learning A-Z™: convolutional neural networks (CNN) – step 1: convolution operation.** Brisbane: SuperDataScience, 2018. Disponível em: <https://www.slideshare.net/KirillEremenko/deep-learning-az-convolutional-neural-networks-cnn-step-1-convolution-operation>. Acesso em: 19 mai. 2019.

GIJSBERS, P.; LEDELL, E.; THOMAS, J; POIRIER, S; BISCHL, B; VANSCHOREN, J. An open source AutoML benchmark. In: 6th INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML), 2019, Long Beach, CA, United States of America. **Proceedings…** Long Beach: ICML, 2019. p. 1-8.

KERAS convolutional layer code. Disponível em: <https://github.com/keras-team/keras/blob/master/keras/layers/convolutional.py#L234>. Acesso em: 03 nov. 2019.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning book**. Boston, AM, United States of America: MIT: MIT Press, 2016.

GROVER, L. K. A fast quantum mechanical algorithm for database search. In: SYMPOSIUM ON THEORY OF COMPUTING, 1996, Philadelphia, PA, United States of America. **Proceedings…** New York: ACM, 1996. p. 212-219.
HAN, K. H.; KIM, J. H. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization, **IEEE Transactions on Evolutionary Computation**, v. 6, n. 6, p. 580-593, 2002.

HARROW, A. W. **Quantum compiling**. 2001. 75 f. Thesis (Bachelor of Science in Physics). Department of Physics, MIT, Massachusetts, 2001.

IBM Q experience. Cambridge: IBM, 2016. Disponível em: https://quantumexperience.ng.bluemix.net/qx/editor. Acesso em: 10 mar. 2019.

KARCZEWSKI, R.; MOZEJKO, M.; SUSIK, M. Inhibited softmax for uncertainty estimation in neural networks. In: 7th INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATION (ICLR), 2019, New Orleans, LA, United States of America. **Proceedings…** New Orleans: ICLR, 2019. p. 1-13.

KHATRI, S.; LAROSE, R.; POREMBA, A.; CINCIO, L.; SORNBORGER, A. T.; COLES, P. J (2019). **Quantum-assisted quantum compiling**. Disponível em: <https://quantum-journal.org/papers/q-2019-05-13-140/pdf/?>. Acesso em: 19 mai. 2019.

KHORSAND, A. R.; AKBARZADEH-T, M. R.; MOIN, H. Genetic quantum algorithm for voltage and pattern design of piezoelectric actuator. In: THE 2006 IEEE CONGRESS ON EVOLUTIONARY COMPUTATION, 2006, Vancouver, BC, Canada. **Proceedings…** Vancouver: IEEE, 2006. p. 2593-2600.

KINGMA, D. P.; BA, J. L. Adam: A method for stochastic optimization. In: 3th INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATION (ICLR), 2015, San Diego, CA, United States of America. **Proceedings…** San Diego: ICLR, 2015. p. 1-15. Disponível em: <https://arxiv.org/pdf/1412.6980.pdf>. Acesso em: 01 mai. 2019.

KITAEV, A. Y. Quantum computation: algorithms and error correction, **Russian Mathematical Surveys**, v. 52, n. 6, p. 1991, 1997.

LECUN, Y.; BOSER, B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; JACKEL, L. D. Backpropagation applied to handwritten zip code recognition, **Neural Computation**, v. 1, n. 4, p. 541-551, 1989.

LEVINE, S.; KOLTUN, V. Learning complex neural network policies with trajectory optimization. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 31. 2014, Beijing, China. **Proceedings…** Beijing: ACM, 2014. p. 829–837

MARKOV, I. L.; SAEEDI, M. Constant-optimized quantum circuits for modular multiplication and exponentiation, **Quantum Information and Computation**, v. 12, n. 5-6, p. 361–394, 2012.

MARIED, E. K., ELDALI, M. A., ZIADA, O. O. **A literature study of deep learning and its application in digital image processing**. Disponível em: < https://www.researchgate.net/publication/317346042_A_Literature_Study_of_Deep_learning_and_its_application_in_Digital_Image_Processing>. Acesso em: 06 jul. 2019.

MITCHEL, T. M. **Machine learning**, New York, NY, United States of America: McGraw-Hill, 1997.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; GRAVES, A.; ANTONOGLOU, I.; WIERSTRA, D.; RIEDMILLER, M. Playing atari with deep reinforcement learning, **Nature**, v. 518, p. 529–533, 2015.

MOR-YOSEF, S.; SAMUELOFF, A.; MODAN, B.; NAVOT, D.; SCHENKER, J. G. Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study. **Obstetrics & Gynecology**, v. 75, n. 6, p. 944, 1990.

NG, A. Y.; KIM, H. J.; JORDAN, M. I.; SASTRY, S. Autonomous helicopter flight via Reinforcement Learning. **Advances in Neural Information Processing Systems**, v. 16, n. 16, p. 363–372, 2004.

NIELSEN, M. A. **Neural networks and deep learning**, Determination press, 2015. Disponível em: <http://neuralnetworksanddeeplearning.com>. Acesso em: 24 out. 2018.

NIELSEN, M. A; CHUANG, I. L. **Quantum computation and quantum information**, Cambridge, Cambridgeshire, United Kingdom: Cambridge University Press, 2010.

PELLEGRINI, J.; WAINER, J. Processos de decisão de Markov: um tutorial. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 133-179, 2007.

PORTUGAL, R.; LAVOR, C. C.; CARVALHO, L. M.; MACULAN, N. Uma introdução à computação quântica, **Notas em Matemática Aplicada**. 8ª ed. São Carlos, SP: Sociedade Brasileira de Matemática Aplicada e Computacional, 2004. p. 62.

PUTERMAN, M. L. **Markov decision processes: discrete stochastic dynamic programming**. New York, NY, United States of America: Wiley-Interscience, 1994.

RIGETTI. Berkeley: RIGETTI, 2019. Disponível em: <https://www.rigetti.com/>. Acesso em: 19 mai. 2019.

RUDER, S. **An overview of gradient descent optimization algorithms**. Disponível em: https://arxiv.org/pdf/1609.04747.pdf. Acesso em: 24 out. 2018.

SHAHRI, A. Z. **Implementation of quantum gate operations using neural networks**. 2017. 69 f. Thesis (Master of Science in Electrical Engineering). Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, Kansas, United States of America, 2017.

SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, **SIAM Journal on Computing**, v. 26, n. 5, p. 1484-1509, 1997.

SILVER, D.; HUANG, A.; MADDISON, C. J.; GUEZ, A.; SIFRE, L.; VAN DEN DRIESSCHE, G.; SCHRITTWIESER, J.; ANTONOGLOU, I.; PANNEERSHELVAM, V.; LANCTOT, M.; DIELEMAN, S.; GREWE, D.; NHAM, J.; KALCHBRENNER, N.; SUTSKEVER, I.; LILLICRAP, T.; LEACH, M.; KAVUKCUOGLU, K.; GRAEPEL, T.; HASSABIS, D. Mastering the game of Go with deep neural networks and tree search. **Nature**, v. 529, p. 484–489, 2016.

STEWART, J. **Cálculo, volume 2**. São Paulo, SP: Cengage Learning, 2013.

TEKNOMO, K. **Q-learning numerical example**. Disponível em: <https://people.revoledu.com/kardi/tutorial/ReinforcementLearning/Q-Learning-Example.htm>. Acesso em: 21 jun. 2019.

WATKINS, C. J. C. H. **Learning from delayed rewards.** 1989. 241 f. Tese (Ph. D.). King's College, Londres, UK, 1989.

ZHANG, Q.; ZHANG, M.; CHEN, T.; SUN, Z.; MA, Y.; YU, B. Recent advances in convolutional neural network acceleration. **Neurocomputing**, v. 323, p. 37-51, 2019.

ZHAO, M.; LI, T.; ALSHEIKH, M. A.; TIAN, Y.; ZHAO, H.; TORRALBA, A.; KATABI, D. Through-wall human pose estimation using radio signals. In: THE IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2018, Salt Lake City. **Proceedings…** Salt Lake City: IEEE, 2018. p. 7356-7365.